HP 3000

MULTIPROGRAMMING EXECUTIVE SYSTEM MANAGER/ SUPERVISOR CAPABILITIES

HEWLETT hp PACKARD

3000 computer systems

HP 3000

MULTIPROGRAMMING EXECUTIVE SYSTEM MANAGER/ SUPERVISOR CAPABILITIES

HP MANUAL PART NO. 03000-90038
MICROFICHE PART NO. 03000-90089

May 1974

List of Effective Pages

Pages	Effective Date
Title	May 1974
Copyright	May 1974
iii to ix	May 1974
1-1 to 1-10	May 1974
2-1 to 2-28	May 1974
3-1 to 3-26	May 1974
4-1 to 4-28	May 1974
5-1 to 5-32	May 1974
6-1 to 6-22	May 1974
7-1 to 7-4	May 1974
8-1 to 8-13	May 1974
9-1 to 9-18	May 1974
A-1 to A-7	May 1974
B-1 to B-4	May 1974
C-1 to C-3	May 1974
D-1 to D-2	May 1974
E-1	May 1974
F-1 to F-5	May 1974
G-1 to G-2	May 1974
H-1 to H-11	May 1974
I-1 to I-6	May 1974
J-1 to J-4	May 1974
K-1	May 1974
L-1	May 1974
M-1 to M-2	May 1974
N-1 to N-2	May 1974
O-1 to O-2	May 1974

Printing History

Manual Edition		Software Version*
First Edition.	Nov. 1972	MPE 32000A.00.00
Second Edition	Sep. 1973	MPE 32000B.00.00
Update Package 1	Jan. 1974	MPE 32000B.00.05
Update Package 2	Mar. 1974	MPE 32000B.00.06
Third Edition	May 1974	MPE 32000B.00.06

*Software update levels not listed in this column do not require a manual change.

Preface

This manual is intended for HP 3000 Computer users assigned the System Manager and/or System Supervisor Capabilities under the Multiprogramming Executive Operating System (MPE/3000, Release B). The manual introduces the main features of the system and provides a general overview of MPE/3000, explaining all significant points that a user with System Manager or Supervisor Capability might need to know. It presents information on how to start-up and shut-down the system, and how to log-on and enter commands in both the interactive and batch-job modes.

For the user with the *System Manager Capability*, the manual tells how to create, change and delete the accounts under which all users access the system, and how to use the system accounting facility. The manual also shows how to assign the Account Manager Capability that allows a user organizational control over an individual account.

For the user with the *System Supervisor Capability*, the manual tells how to generate and modify MPE/3000, display job status information, manage the MPE/3000 scheduling queue and system logging facility, and permanently allocate code in virtual memory.

This manual is a reference book rather than a tutorial text for new programmers. The reader should understand the basic operating principles of the HP 3000 Computer. He should also examine the following manual for an overview of the interrelationships between the main hardware and software features offered:

Title	Part No.
HP 3000 Software General Information Manual	*03000-90001*

The user may also want to read the following manuals for further information on MPE/3000, the HP 3000 Systems Programming Language (SPL/3000), and the functions available through the MPE/3000 Operator's Console.

Title	Part No.
HP 3000 Systems Programming Language	*03000-90002*
HP 3000 Systems Programming Language Textbook	*03000-90003*
HP 3000 Multiprogramming Executive Operating System	*03000-90005*
HP 3000 Multiprogramming Executive Console Operator's Guide	*03000-90006*

Contents

FIGURES

SECTION I
Introduction to MPE/3000

The Multiprogramming Executive Operating System (MPE/3000) is a general-purpose, disc-based software system that supervises the processing of user programs submitted to the HP 3000 Computer. MPE/3000 relieves the user from many program control, input/output, and other housekeeping responsibilities by monitoring and controlling the input, compilation, run preparation, loading, execution and output of user programs. MPE/3000 also controls the order in which programs are executed, and allocates the hardware and software resources they require.

FEATURES

MPE/3000 offers the user many important features; some of these are found elsewhere only in medium to large-scale computers.

Multiprogramming

Through multiprogramming (interleaved processing), MPE/3000 allows numerous users to execute many different programs concurrently. The number of programs that can be processed concurrently depends on such factors as the hardware configuration, program operating modes (batch or interactive), and applications involved. Each programmer, however, uses the computer as if it were his own private machine — in other words, he need not depend on, nor even be aware of, others using the machine.

General-Purpose Versatility

MPE/3000 allows users to run batch and interactive programs concurrently.

BATCH PROCESSING. Batch processing lets programmers submit to the computer, as a single unit, commands that request various MPE/3000 operations such as program compilation and execution, file manipulation, or utility functions. Such a unit is called a *job.* Jobs contain all instructions to MPE/3000 and references to programs and data required for their execution; once a job is running, no further information is needed from the programmer. (Frequently, new programs and data are submitted as part of the job.)

Jobs are input through batch input devices such as card readers. In fact, several jobs can be submitted from multiple batch input devices concurrently. MPE/3000 schedules each job according to its priority. When a job enters execution, the commands within it are sequentially executed on a multiprogramming basis. MPE/3000 generates the job output on a local device such as a line printer, tape unit, or disc unit, or on a local or remote terminal. When one job is temporarily suspended, perhaps to await input of data, another (if available) immediately enters execution. Thus, when many jobs are active in the system, continuous processing and high throughput can be maintained.

INTERACTIVE PROCESSING. In interactive processing, the programmer interacts conversationally with the computer, receiving immediate responses to his input. Many users at different remote or local terminals can program on-line in this fashion. This type of interaction, called a *session*, can be used for program development, information retrieval, computer-assisted education, and many more applications where the user at a remote terminal must access the system directly.

MPE/3000 can continue to execute batch jobs at the same time it handles sessions. The basic difference between a session and a batch job is that a session is interactive, but a job is not. Thus, during a session, the user maintains a dialogue with the system to control input and monitor output; in a batch job, however, the command stream is entered into the system without a user/system dialogue.

Choice of Programming Languages

Under MPE/3000, the user can submit programs written in the following languages:

- FORTRAN/3000
- BASIC/3000
- COBOL/3000
- SPL/3000 (Systems Programming Language)

In addition, MPE/3000 supports subsystems for editing program files, performing statistical operations, aiding in debugging programs, running on-line diagnostic programs, and various other applications.

Each language translator and subsystem is accessed by a unique MPE/3000 command. The programmer need learn only one set of conventions for using these programs, because they all use the same general command formats, special characters, and error-diagnostic methods. Command coding is based upon the American Standard Code for Information Interchange (ASCII) Character Set, shown in Appendix A.

Operating Simplicity

MPE/3000 is easy to initialize, operate, monitor, and shut down. Its operation is overseen by a user with the MPE/3000 System Manager capability, who assigns programming capabilities of various levels to each user. *Standard capabilities* allow the typical "general applications" programmer to interact with the computer through a batch input device or a terminal. If this programmer is planning only to compile and execute a batch job, or to run an on-line interactive program, he may need to know only a few simple MPE/3000 commands. As his needs become more extensive, however, so must his knowledge of MPE/3000. A user planning more complex operations can be assigned various *optional capabilities.* These allow him to access more sophisticated system resources for such tasks as executing privileged instructions. Sets of capabilities are also used to protect the system and its users by limiting access to special system capabilities only to those who understand their correct use. Capability sets greatly simplify use of the system from the stand-point of each individual user — they define the extent to which he must understand and interrelate with MPE/3000, and permit a user to ignore aspects of MPE/3000 that do not apply to him. (Standard and optional capabilities are described in detail in *HP 3000 Multiprogramming Executive Operating System (03000-90005).*)

Input/Output Conveniences

Because MPE/3000 treats all input/output devices as files (or groups of files in the case of mass storage devices), the programmer may access these devices by file names rather than by device types or logical unit numbers. The file names used in programs are independent of the devices used for file input and output, and need only be associated with these devices at the time the programs are run. This means that the user can write programs without immediate concern for the physical source of input or destination of output. This independence also means that programs can be run in either batch or interactive mode without changing the names of the files they reference.

Files on disc can be structured for either direct or sequential access, on an exclusive or shared basis. Direct-access files contain fixed-length records; sequential files, however, can contain either fixed- or variable-length records. For files on disc, storage space is automatically allocated as it is needed. MPE/3000 permits simultaneous access of sharable disc files by many programmers.

Processing Efficiency

MPE/3000 offers the user increased throughput by taking best advantage of the HP 3000 Computer architecture and resources, and the rapid speed of the central processor. This, in turn, permits the rapid changes between user environments that make multiprogramming in batch and interactive modes possible.

RE-ENTRANT CODE AND PRIVATE DATA. Within MPE/3000, many user and system functions can be active simultaneously without mutual interference. This is because the hardware provides protection of programs and guarantees the privacy of user data areas. MPE/3000 keeps code and data logically separate by organizing them into re-entrant code segments (which can be shared among users but not altered) and data segments (which cannot be shared but which can be altered by the creating user). *Code-sharing* means that only one copy of each program, accessible to many users concurrently, need exist in memory at any time. *Code segmentation* allows code to be moved from disc into main memory only when needed and dynamically relocated in main memory to accommodate other programs and routines. *Code re-entrancy* means that when a program is interrupted during execution of a code segment, and another program uses that same segment, the segment is completely protected from modification and will be returned, intact, to the previous program.

STACK ARCHITECTURE. Many powerful operating system features are made possible by the computer's use of stacks (linear storage areas for data) where the last item stored in is always the first item taken out. Some of these features are

- Ease of compilation and parameter passing
- Fast execution
- Highly-efficient subroutine linkage
- Minimum overhead
- Dynamic allocation of temporary storage
- Rapid interruption and restoration of user environments
- Code compression
- Recursion (where a procedure calls itself)

MICROPROGRAMMING AND THE CENTRAL PROCESSOR. Additional economy has been provided by microprogramming many system operations normally provided by software. These operations are requested by machine instructions that each, in turn, execute many microinstructions built into the central processor hardware. Microprogramming eliminates the repetitive coding and main memory requirements otherwise needed for recurring operations (such as moving character strings from one location to another, or scanning strings for a particular character), thereby placing the operating burden on the highly-efficient microprocessor rather than on MPE/3000 software.

VIRTUAL MEMORY. The MPE/3000 virtual memory offers users a total memory space that far exceeds the maximum main-memory size of 131,072 bytes. Virtual memory consists of both main memory and an extensive, flexible storage area on disc. User programs and information in the disc area are subdivided into units (segments) of code or data that are dynamically moved, through overlays, into main-memory (core) for execution.

HIGH DATA THROUGHPUT. High throughput of data is facilitated by high-speed channels, double-buffering, many input/output devices, and input/output program execution (through an input/output processor) in parallel with regular program execution (by the central processor).

AUTOMATIC SCHEDULING. MPE/3000 automatically schedules all jobs and sessions according to their priorities. When execution of a running program is interrupted for any reason (such as input/output or an internal interrupt), MPE/3000 passes control to the program of highest priority awaiting execution.

Accounting Facility

MPE/3000 keeps track of various system resources used by each job/session, group, and account; these resources include permanent file space, central-processor time, and (for sessions) terminal connect time. Limits can be set for the maximum use of these resources at the group or account level. As each job/session is logged off, the resource-use counters are updated. When another job/session attempts to log-on, and the central processor or (for sessions) terminal connect time limits have previously been exceeded, access is refused. When a request is made to save a file, and this action would result in exceeding the permanent file space limit at either the account or group level, the request is denied.

The accounting information for each group and account can be extracted and displayed, showing counts and limits for permanent file space (in disc sectors), central processor time (in seconds), and connect-time (in minutes).

Logging Facility

MPE/3000 enables a user with System Supervisor Capability to control the recording (on disc) of information about overall system activity. This log can be displayed through user-created analysis routines. The user can specify the information to be recorded on the log file, making his selection from the following areas: job/session and program initiation and termination, file closing, and system start-up and shut-down.

System-to-System Compatibility

All HP 3000 Computers operate under a single operating system — MPE/3000. This means that programs prepared on one HP 3000 can be run on any other without modification (provided that all devices required by those programs are connected on-line). It also means that users moving from one installation to another need not undergo additional training or read additional documentation to prepare themselves for the new environment.

Program and File Security

Each user operates in an environment protected from interference by other users. Program protection is provided by the hardware; file security is provided by a software facility based upon a series of lockwords and hierarchical access restrictions that allow the programmer to specify the degree of security desired.

System Manager and System Supervisor Capabilities

MPE/3000 provides many tools for overall management and control of the system to persons with the *System Manager* and *System Supervisor Capabilities.*

The System Manager Capability allows a user to have final control of the overall use of the system by defining the accounts under which users access MPE/3000, and the resource-use limits (if any) that apply to these accounts. It also allows a user to assign the *Account-Manager Capability* that permits its possessor to name the users who can access an individual account, and to partition the file domain of that account into groups of files.

The System Supervisor Capability allows a user to supervise and control the general operation of the system by:

- Creating tapes for backing-up and modifying the system.
- Displaying certain system information.
- Permanently allocating/deallocating programs in virtual memory.
- Exercising greater scheduling control over processes than that allowed other users.
- Managing the system log file.

It is important to bear in mind that the above capabilities are granted to users through the assignment of special *attributes*, and in no way imply formal responsibilities or duties. Thus, a regular programmer may have the System Manager Capability or the System Supervisor Capability, or both capabilities. For this reason, it is best to speak of "a user with System Manager Capability" rather than a formal "System Manager."

SOFTWARE

MPE/3000 is furnished to the customer on a reel of magnetic tape. As part of the operating system software, these major components are provided:

- *System Configurator*, for configuring and making a back-up copy of MPE/3000.
- *System Initiator*, for installing and starting MPE/3000.
- *Command Interpreter*, for handling the commands that allow the user to interact with MPE/3000 through a terminal or batch input device.
- *File Management System*, for providing uniform access to disc files and standard input/output devices, and maintaining file security.
- *Memory Management System*, for dynamically allocating main-memory among contending users.
- *Dispatcher*, for allocating central-processor time among programs in execution.
- *Input/Output System and Drivers*, for scheduling, initializing, monitoring, and completing input/output requests for standard devices (accessed through the File Management System).
- *System Library*, for storage of frequently-used routines sharable among many users.

- *Segmenter Subsystem*, for segmenting and loading programs, and resolving references to external code.
- *Accounting System*, for maintaining and displaying resource usage counts.
- *Logging System*, for maintaining the system log file.

No additional or auxiliary software is required to install, operate, or maintain MPE/3000.

HARDWARE

The minimum hardware configuration required by MPE/3000 is

- Mainframe and Accessory Equipment Supplied, including HP 3000 Central Processor and 65,536 Bytes of Main Memory (Core), SIO Multiplexer Channel, System Control Desk (with System Console), and Asynchronous Terminal Controller
- Disc
- Magnetic Tape Unit

The disc is used to contain portions of MPE/3000 and user programs and data. The magnetic tape unit is used for cold-loading MPE/3000, loading subsystems (such as compilers), and storing user programs and data. (MPE/3000 is initialized and maintained through the console, which can also be used for input/output of user programs.)

The following optional hardware can be added to the system:

- Main Memory (for a total of up to 131,072 bytes)
- Discs (Fixed-Head or Moving-Head)
- Magnetic Tape Units
- Card Readers
- Line Printers
- Card Punches
- On-Line Terminals
- Paper Tape Readers
- Paper Tape Punches
- Selector Channels

With this optional equipment, many hardware configurations are possible. For example, a small system used for batch processing might appear as shown in Figure 1-1. In this system, the card reader is used for input and the line printer for output, while the disc is used for storing system and user programs and data. The magnetic tape unit is used for cold-loading MPE/3000 and for storing user programs and data. The console is used to initialize and maintain MPE/3000.

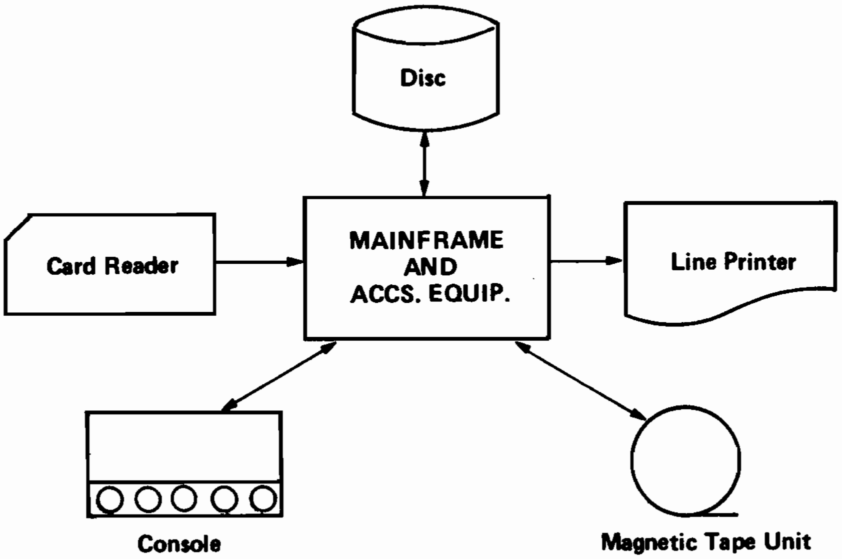


Figure 1-1. Small Batch System

A small interactive system might be configured as shown in Figure 1-2. In this system, up to seven terminals are used for input and output.

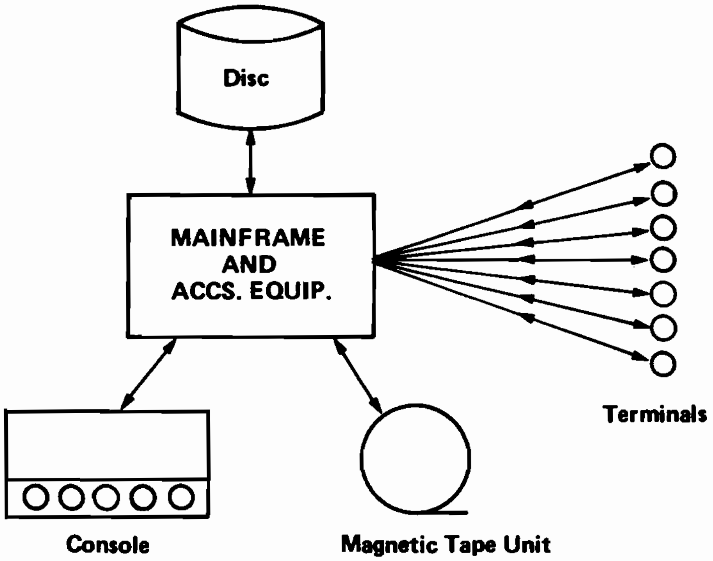


Figure 1-2. Small Interactive System

A combined batch and interactive system is illustrated in Figure 1-3. Here, batch and terminal operations can occur separately or concurrently.

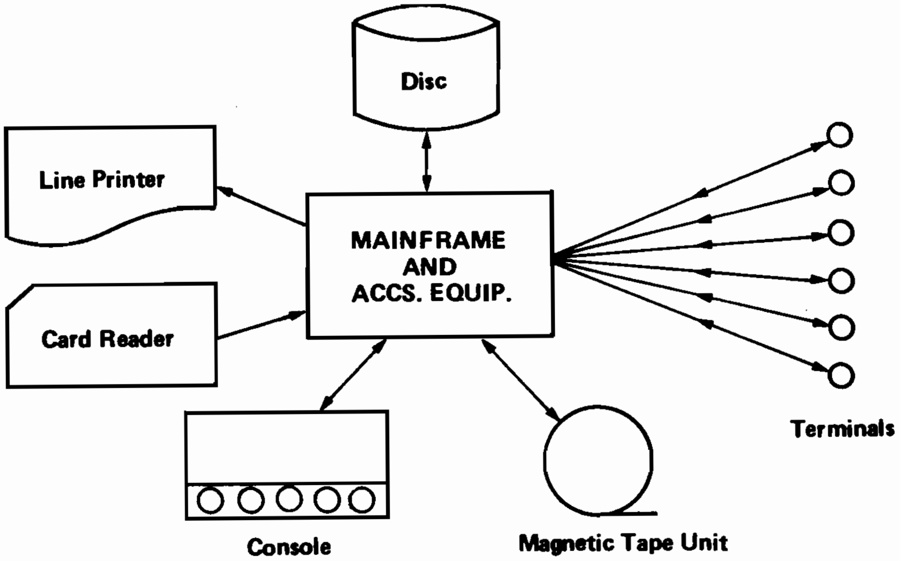


Figure 1-3. Combined Batch and Interactive System

A large processing system is shown in Figure 1-4. This system incorporates a large central memory, fixed-head and moving-head discs, many input/output devices for multiple-job batching, and eight terminals for remote processing.

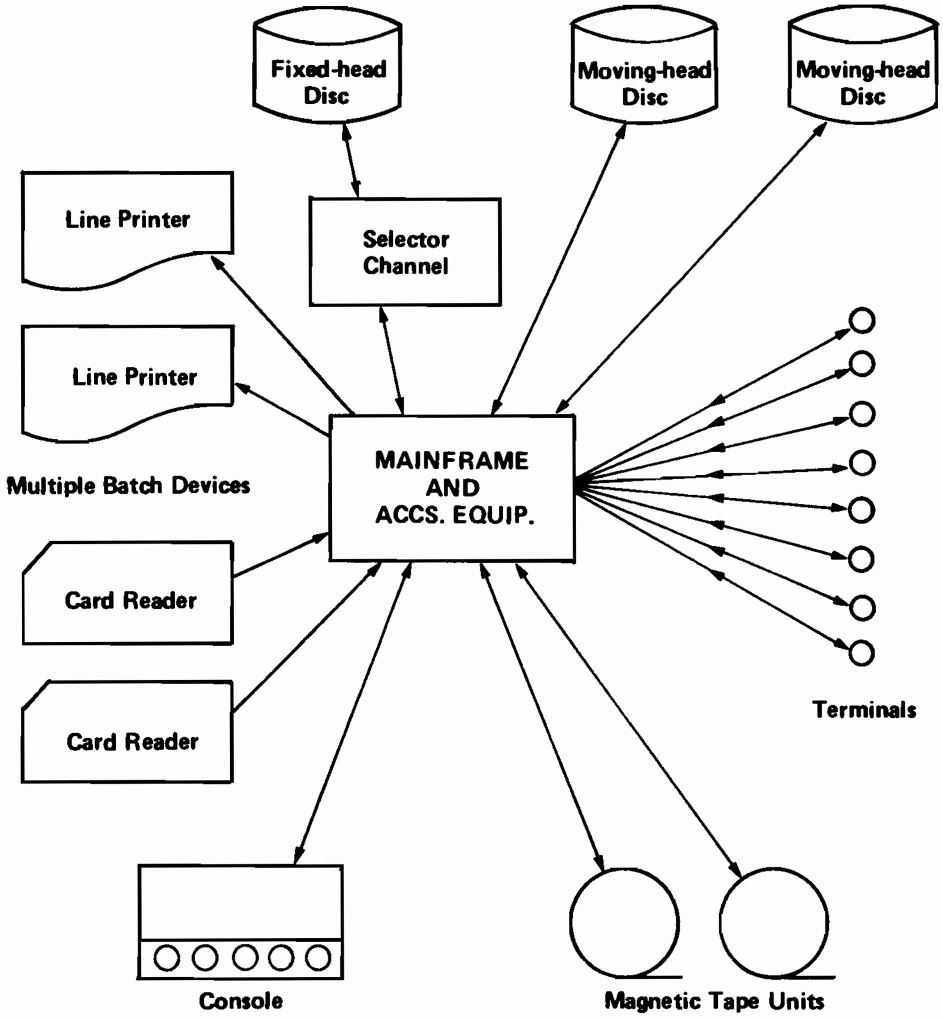


Figure 1-4. Large Processing System

SECTION II
How MPE/3000 Operates

In supervising the compilation and execution of user programs, MPE/3000 performs the following tasks:

- Controls all input/output.
- Allocates memory to user programs and the system routines required for their support.
- Schedules batch programs for access to the central processor.
- Allocates central processor time to programs.
- Swaps portions of programs to and from main memory as required during execution.

NOTE: For users employing the MPE/3000 System Manager, System Supervisor, and standard capabilities only, MPE/3000 automatically performs these and most other functions at their command. Generally, such users need not be concerned with the internal aspects of MPE/3000 discussed in the first part of this section, and may skip ahead to the heading ACCOUNT/GROUP/USER ORGANIZATION for a discussion of their relationship to the system. However, the information in the beginning of this section will be of value to users with the System Supervisor Capability who plan to create and run processes; they should read it carefully.

MPE/3000 RESIDENCE

MPE/3000 resides in three general areas:

1. Those routines most frequently used reside in *main memory*, all of which is part of virtual memory. (Routines not residing in main memory are moved there only when required. This ensures that as much main memory as possible remains available for the execution of user programs.)

2. Frequently-used routines reside on *portions of disc within virtual memory;* these disc areas are treated as logical extensions of main memory.

3. All remaining routines (including a permanent copy of each compiler) are stored in *areas of disc outside of virtual memory;* these areas are treated as conventional secondary storage.

PROCESSES

In MPE/3000, programs are run on the basis of *processes* created and handled by the system. A process is not a program itself, but the unique execution of a program by a particular user at a particular time. Therefore, if the same program is run by several users, or more than once by the same user, it is used in several distinct processes.

The process is the basic executable entity in MPE/3000. A process consists of a process control block that defines and monitors the state of the process, a dynamically-changing set of code segments, and a data area (stack) upon which these segments operate. (Thus, while a *program* consists of instructions (not yet executable) and data in a file, a *process* is an executing program with a data stack assigned.) The code segments used by a process can be shared with other processes, but its data stack is private (Figure 2-1). For example, each user working on-line through the BASIC language is running his program under a separate process; all use the same code (the only copy of the BASIC interpreter in the system), but each has his own stack. (The interrelationship of the process control block, code segments, and data stack is discussed in detail under "PCB/Code Segment/Stack Interaction.")

Processes, and the elements that comprise them, are invisible to the programmer accessing MPE/3000 through standard capabilities; in other words, this programmer has no control over processes or their structure. For this user, MPE/3000 automatically creates, handles and deletes all processes. The user with certain optional capabilities, however, can create and manipulate processes directly.

The entire MPE/3000 system is a collection of processes that are either active or dormant. These processes are logically organized in a tree structure (see Figure 2-2); each process has only one immediate ancestor (father) but could have several immediate descendants (sons). The root process that controls all other processes in MPE/3000 (Figure 2-2) is the only process that has no ancestor. It is the first process established, and is created during system initialization. During system configuration, the root process uses the data specified to create various system processes, including the input/output and user controller processes; the user controller process creates its own descendent processes. (A *system process* is a process that executes on behalf of MPE/3000.)

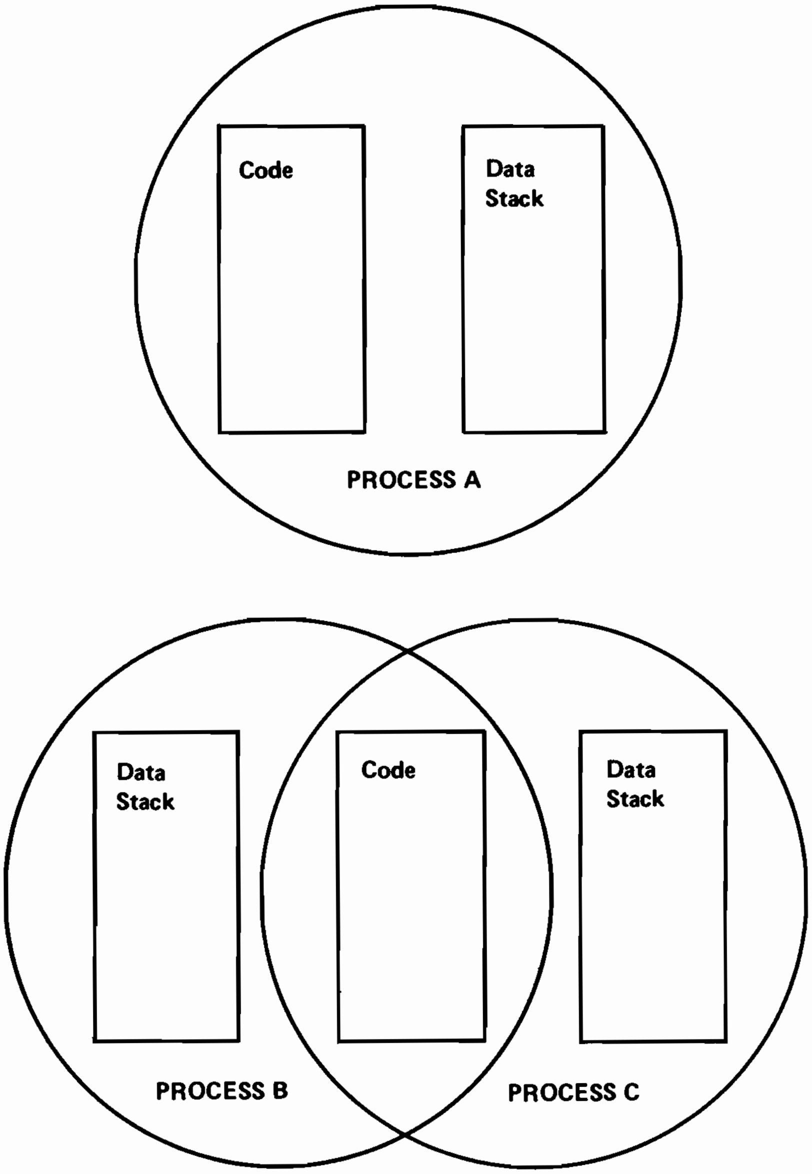


Figure 2-1. Code-Sharing and Data Privacy

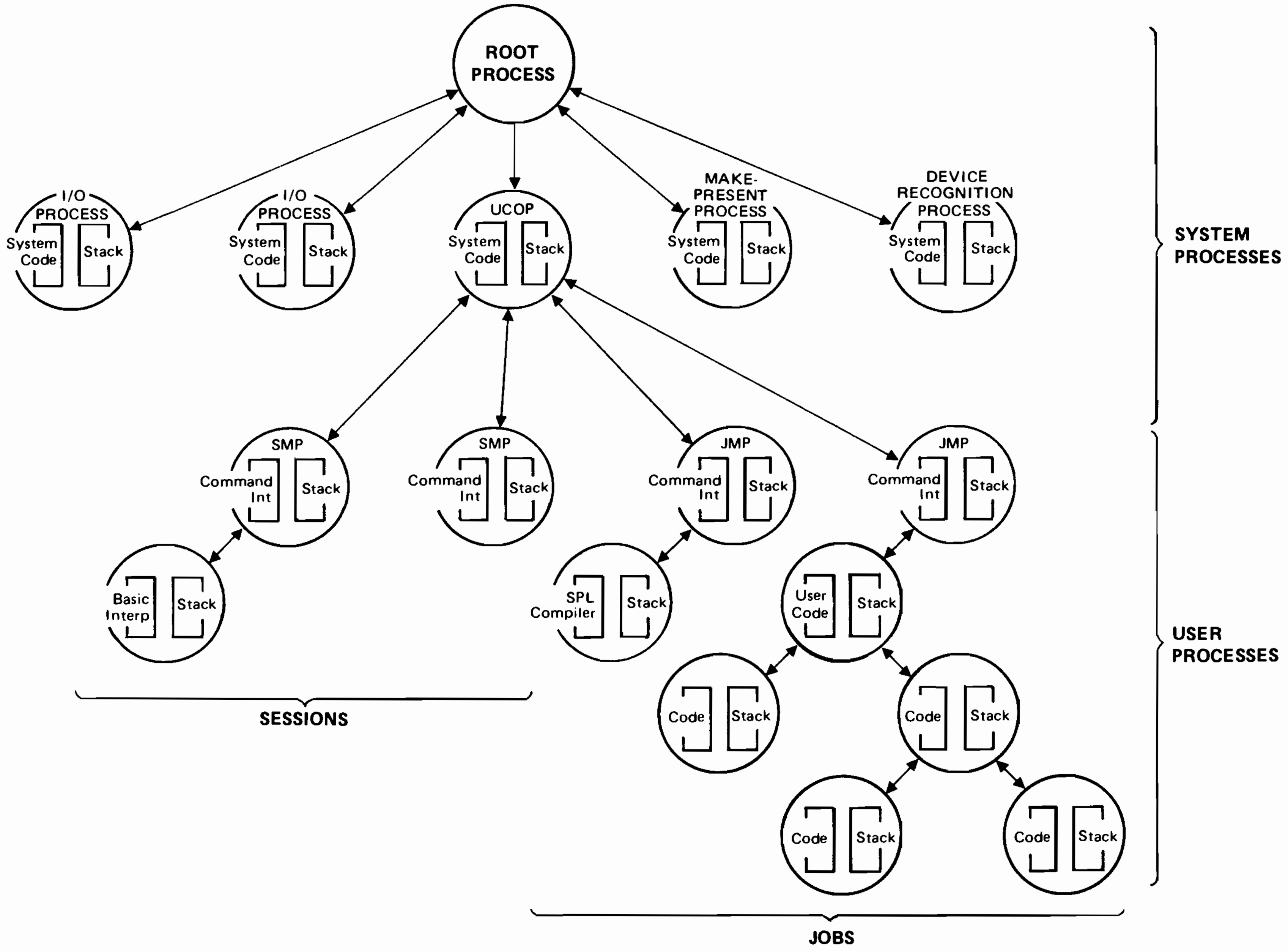


Figure 2-2. Process Organization

The user controller process (UCOP) is the ancestor of all existing user processes. (A *user process* is a process that runs on behalf of the user.) When the user first accesses MPE/3000, the UCOP creates a *main process* that oversees the execution of his program. If the user is running a batch program (called a *job*), a job main process (JMP) is created. If he is executing an interactive program (*session*), a session main process (SMP) is created.

All JMP's and SMP's use system code, the MPE/3000 Command Interpreter, interfacing directly with the user. They may create descendants that use system code, such as the BASIC/3000 Interpreter or FORTRAN/3000 or SPL/3000 compilers when translating source programs; when executing user programs, of course, the descendant processes use the user's code.

PROCEDURES

In MPE/3000 most individual programming operations are handled by unique sets of code known as *procedures.* (Others are handled by special programs or interrupt routines.) In SPL/3000, the language in which MPE/3000 is coded, a procedure is defined by a procedure declaration consisting of

- A procedure head, containing the procedure name and type, parameter definitions, and other information about the procedure

- A procedure body, containing executable statements and data declarations local to this procedure

As part of their function, many procedures also return parameter values to the processes that invoke them.

In SPL/3000, each procedure is executed by a corresponding *procedure call.* When a procedure call is encountered, control is transferred to the procedure, which runs until an exit is encountered. Control is then returned to the statement following the call for that procedure.

While a procedure in SPL/3000 is similar to a subroutine in many respects, there are major differences between the two:

- When referencing a logical set of code, a subroutine call is significantly faster and more efficient than a procedure call, while a procedure call has more powerful (and general) applications.

- A subroutine declaration may be used within a procedure declaration, but a procedure declaration may not appear in a subroutine.

- A procedure allows dynamic storage allocation, permitting local variables.

- A subroutine can contain non-local references to variables declared within the scope of the procedure in which it occurs.

- A subroutine can be invoked only within the procedure of which it is a part.

Unlike procedures, subroutines are not suitable for solving complex problems that require explicit stack manipulation or many temporary storage locations. They *are* useful, however, for various utility and arithmetic-function routines.

In addition to the procedures provided by MPE/3000, SPL/3000 allows the user to write procedures to suit his own purposes. To distinguish MPE/3000 system procedures (which are always available to the user, directly or indirectly) from any other procedures, the term *intrinsics* is applied to MPE/3000 procedures. Similarly, the term *intrinsic calls* is used to denote the procedure calls that reference MPE/3000 procedures. From this point on, these terms will carry these respective meanings throughout this manual.

Each large functional unit of MPE/3000, such as the Command Interpreter, File Management System, or Input/Output System, consists of many intrinsics. The creation, handling, and deletion of processes is performed exclusively through intrinsics not directly accessible by users with the standard MPE/3000 capabilities. (However, these intrinsics can be accessed directly by users having certain optional capabilities described in *HP 3000 Multiprogramming Executive Operating System (03000-90005).)*

SYSTEM LIBRARY

The MPE/3000 system library is a collection of frequently-used routines that can be readily shared among many users. A library code segment consists of one or more procedures. Some of these segments reside permanently in virtual memory, while others are stored elsewhere on disc and called into virtual memory as needed.

USER INTERFACE

In requesting system functions, the programmer never interacts directly with the computer hardware. Instead, he interacts with a dual-level software interface that accepts *commands* (for general functions external to his program) and *intrinsic calls* (for specific functions within his program). In general, the commands and intrinsic calls themselves access uncallable intrinsics that reference machine instructions. These, in turn, execute micro-instructions from the microcode hardwired into read-only memory (Figure 2-3). It is this microcode only that operates directly on the computer hardware.

Commands

Commands are used to initiate and terminate jobs, re-specify file characteristics, compile and execute programs, call various utility subsystems, and perform other broad functions external to user programs. A user with System Manager Capability uses them to create and alter accounts; a user with System Supervisor Capability uses them to display information, back-up and modify the system, and manage the System Log File and scheduling queue. Commands are entered through any standard input device, typically the card reader (for jobs) or the terminal (for sessions). When the input device is connected on-line to MPE/3000, a main process is automatically initiated that uses the MPE/3000 Command Interpreter program to read and translate the commands. The first command entered must always be one that initiates a job or session.

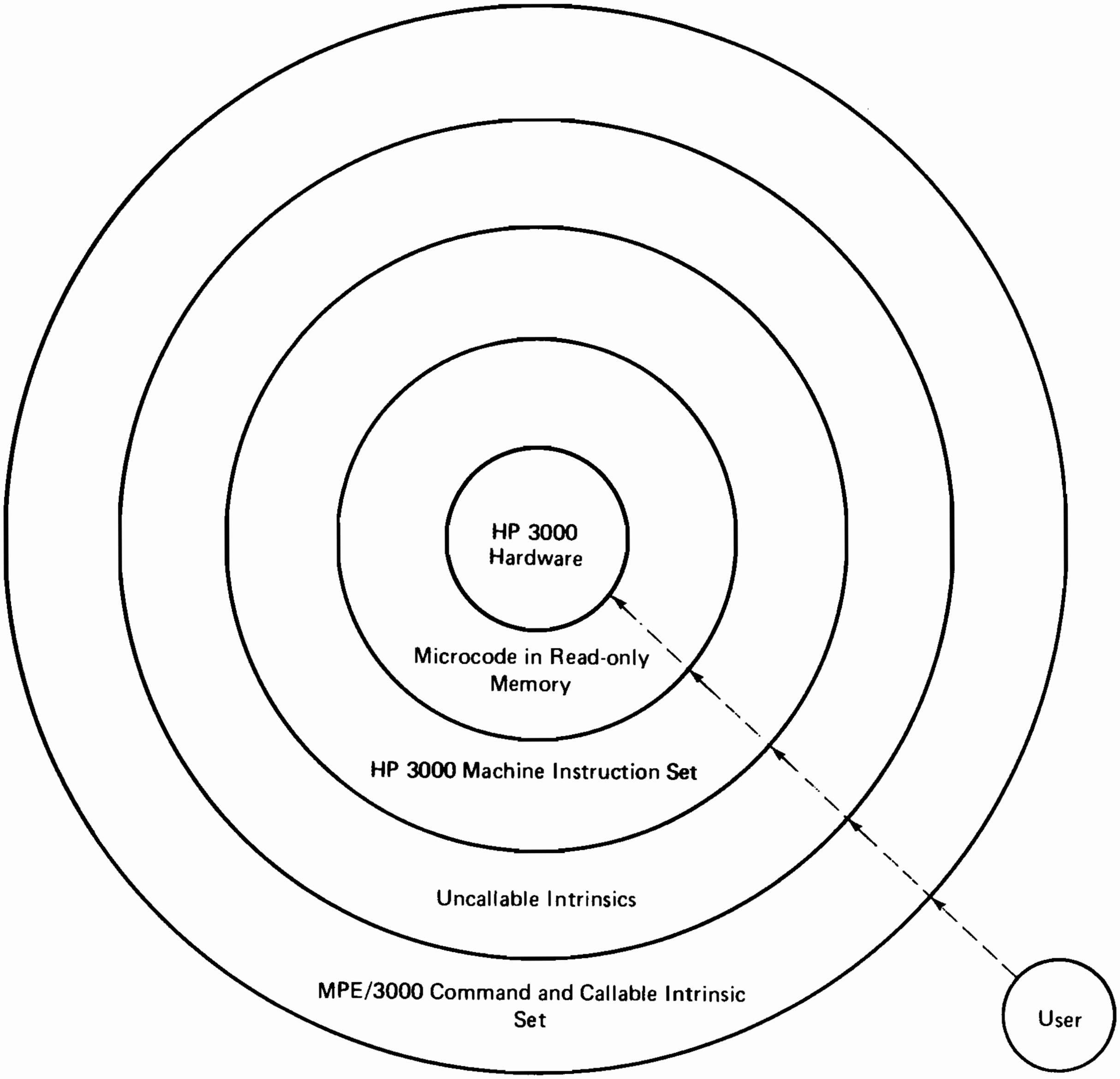


Figure 2-3. User/Software/Hardware Interface

Each command invokes an intrinsic that implements the function desired. (Such intrinsics are called *command executors.*) When a command is entered, the Command Interpreter checks it against the names in the MPE/3000 command dictionary. If the command is valid, the Command Interpreter passes the command parameter list image to the appropriate executor for execution. When the action requested by the command is satisfactorily completed, control returns to the Command Interpreter.

Intrinsic Calls

Intrinsic calls implement MPE/3000 functions requested *programmatically* (within a user's program), such as reading, writing on, and updating files; skipping forward and backward on files, or returning system table information to the user's program. In an SPL/3000 program, the user writes the intrinsic calls explicitly, but in FORTRAN, COBOL, or BASIC programs, in most general applications, the compiler generates any necessary intrinsic calls automatically — they are invisible to the user. (Provisions are also available in FORTRAN, COBOL, and BASIC that allow the user himself to call intrinsics, at his option.) All MPE/3000 intrinsics are treated as external procedures by user programs. External linkages from user programs to intrinsics are satisfied when the user programs are segmented and allocated residence in virtual memory.

The set of intrinsics available for a user depend upon the capabilities assigned to him by the user who is managing the account under which he accesses MPE/3000. For example, only the user with the *Process-Handling Optional Capability* can directly access intrinsics that create and manipulate processes. The set of intrinsics available for a user defines the computer for that user. The user calls *callable* intrinsics appropriate for a particular application, and these intrinsics check the validity of the requests. Callable intrinsics, in general, invoke *uncallable* intrinsics to actually perform the necessary operations. (Uncallable intrinsics are those not available to standard users because their mis-use can be hazardous to the system; they can be accessed, however, by users with a special MPE/3000 optional capability.)

Certain programs that may be executed by all users sometimes require special capabilities. In this case, the program (not the user) is assigned these capabilities by the user who creates the program, as discussed later.

INPUT/OUTPUT

The MPE/3000 Input/Output System schedules, initiates, and completes all input/output requests for all HP-supported devices connected to the system. Normally, the input/output system remains invisible to the user, since he accesses it indirectly through the MPE/3000 File Management System.

The Input/Output System includes an independent controller for each type of hardware device connected to the computer. Each controller is associated with an input/output monitor process. This process services the requests for the controller and calls input/output initiator and completion driver routines as required. Requests to the Input/Output System are issued by the File Management and Memory Management Systems.

File Management System

The File Management System provides a uniform method of directing input and output of information. It handles various input/output applications, such as the transfer of information to and from user processes, compilers, and data management subsystems.

MPE/3000 treats each set of input or output information as a set of logical records called a *file*. When a file is created, MPE/3000 allocates (or requests the operator to allocate) a device for its storage. Input read from a hardware device such as a card reader is accepted directly from that device. Similarly, output from an executing process is transmitted directly to the required device (such as a line printer).

Information is moved between a file and main memory in *physical records*; a physical record is the basic unit that can be transferred to or from the device on which the file resides. In files on disc or magnetic tape, physical records are organized as *blocks* of logical records; the block size is specified by the user, but input/output and blocking are performed by MPE/3000, freeing the user from the actual record-handling details. On unit record devices, however, the physical record size is determined by the device itself, and logical records are not blocked.

The user references a file by the *file name* assigned to the file when it is created. When he does this for an existing file, MPE/3000 determines the device or disc address where the file is stored and accesses the file for him. Throughout its existence, the file remains device-independent.

The MPE/3000 File Management System allocates devices for the storage of files on the basis of specifications from the user. For example, the user can request a device by generic type name (such as any magnetic tape unit or line printer), or by the logical device number that refers to a specific device. Logical device numbers are related to all devices when MPE/3000 is configured.

Scheduling

The MPE/3000 Scheduling System ensures that all processes competing for the central processor access it in an orderly manner. All processes are placed in a master scheduling queue in order of their priority. (This queue is actually a linked list of PCB's — process control blocks — for the processes scheduled.) When a process in execution is completed, terminated, or interrupted, the MPE/3000 Dispatcher program searches the master queue for the process of highest priority awaiting execution and transfers control to that process.

Within the master queue, five standard subqueues are used for scheduling processes created by user programs (Figure 2-4). Since MPE/3000 schedules each process independently, not all processes for a job are necessarily entered in the same subqueue.

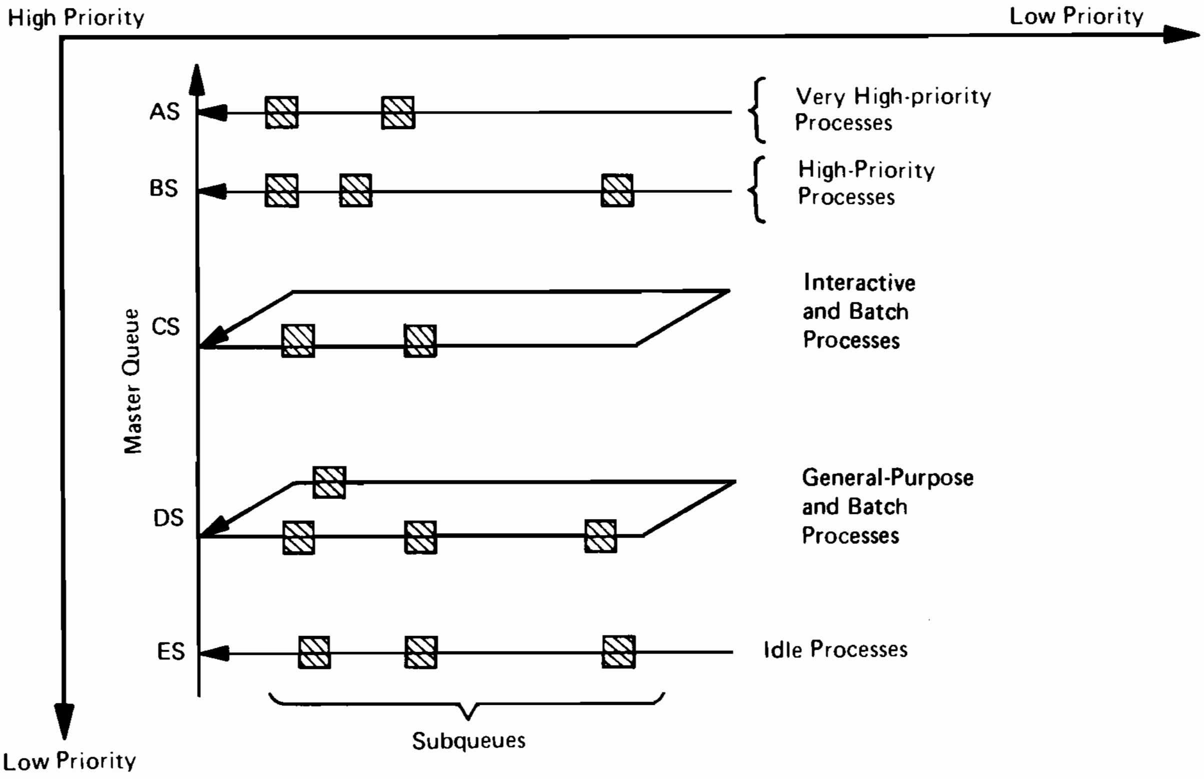


Figure 2-4. Scheduling Queues

Three of the standard subqueues are linear in structure; two of these (AS and BS) are high-priority subqueues that are presently available for general purposes, but in subsequent releases of MPE/3000 will be used for processes with new MPE/3000 optional capabilities; the third linear subqueue (ES) is available for idle processes with low priority. In a linear subqueue, the process with highest priority accesses the central processor first and maintains this access until the process either is completed or suspended to await input/output. (The master queue itself is a linear queue.)

The other two standard subqueues are circular subqueues. One (CS) is used for interactive sessions and multiprogramming batch jobs. The other (DS) is available for general use at a lower priority than the CS subqueue. In these subqueues, each process accesses the central processor for an interval (time-slice) of limited duration. At the end of each time-slice, or when the currently-running process enters a waiting state, control is transferred to the next process in the subqueue, continuing in a round-robin fashion. This time-slicing is controlled by a system timer.

COMPILATION

User programs are entered into the computer in a source-language, translated into binary form by a process executing a compiler, and stored on disc (Figure 2-5A). Because the code making up a compiler is re-entrant, it can be shared by many users. Therefore, only one copy of a compiler is required in memory no matter how many programmers need it at one time to compile their programs.

MPE/3000 regards user source-language programs as consisting of program units. A program unit is the smallest divisible part of any program or subprogram. For example, in SPL/3000, this is an outer block program unit or a procedure; in FORTRAN/3000, this is a main program, subroutine, function, or block data unit. In COBOL/3000, this is a main program, subroutine, or section. Thus, a *program* consists of one or more program units, one of which must be an outer block (SPL/3000), or main program (FORTRAN/3000 or COBOL/3000) program unit. A *subprogram*, which is the smallest individually-compilable entity, in turn, consists of one or more program units.

When a source-language program is compiled, each program unit is transformed into a relocatable binary module (RBM) that contains user code plus header information that labels and describes this code.

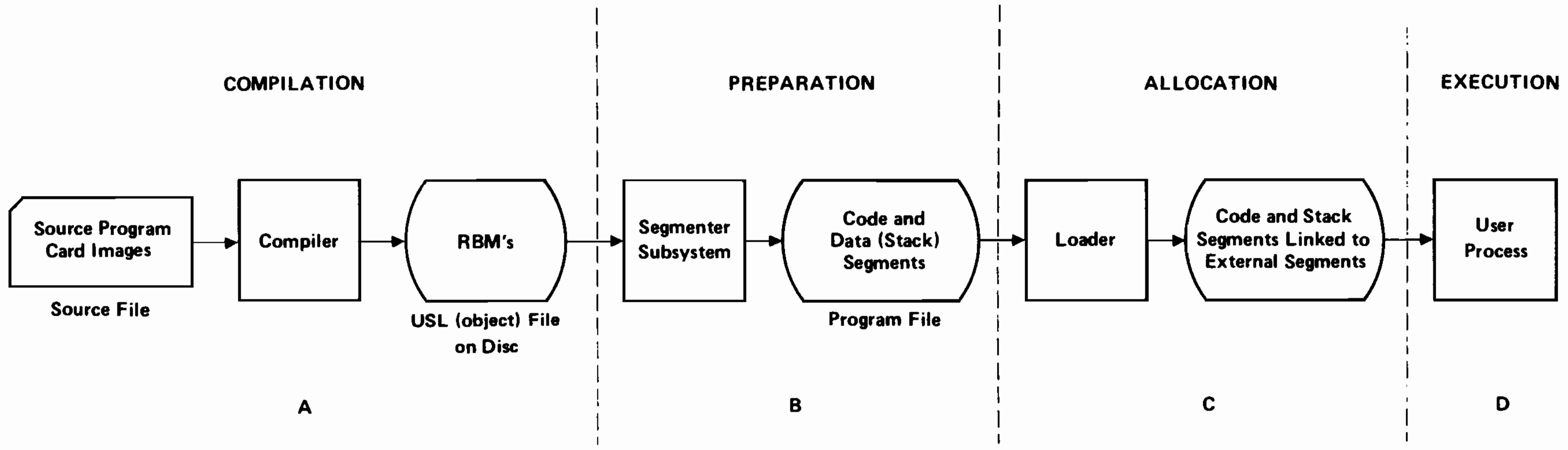

COMPILATION
PREPARATION
ALLOCATION
EXECUTION
Source Program Card Images
Source File
Compiler
RBM's
USL (object) File on Disc
Segmenter Subsystem
Code and Data (Stack) Segments
Program File
Loader
Code and Stack Segments Linked to External Segments
User Process
A
B
C
D

Figure 2-5. Program Management

PROGRAM PREPARATION

The RBM's that result from compilation of a user's program onto disc make up a user subprogram library (USL) file. For each program unit compiled, there is one RBM in the USL.

The USL is not executable, however. Instead, it must be *prepared* for running by MPE/3000. During preparation, a process running the MPE/3000 Segmenter binds the RBM's from the USL (object file) into linked code segments arranged in a *program file* (Figure 2-5B). Each segment contains machine instructions produced from the user's program, plus linkages to other segments. The code in a segment cannot be altered because it is treated as read-only information; thus, it remains re-entrant and can be executed repeatedly by many users. When a program is segmented, a special segment for the input of user data is also initially defined; this contains the stack.

ALLOCATION/EXECUTION

When a program is run, it is allocated and executed. In *allocation*, a process running the MPE/3000 loader binds the segments from the program file to referenced external segments from a library (Figure 2-C). MPE/3000 then creates a process to run the program (Figure 2-D).

Execution now begins. As it progresses, many new processes may be created, run, and deleted. For each process in execution, one or more code segments and one stack, operating under control of a process control block (PCB), typically exist in main memory. Not all code segments belonging to a process in execution need exist in main memory simultaneously. Typically, a process operates on a set of segments that are dynamically swapped between main memory and disc. MPE/3000 maintains records of the frequency each segment is used, so that those used least frequently in main memory become the most eligible for swapping out. The user never needs to determine whether a segment is in main memory or on disc at any given time — this is always done for him automatically by MPE/3000.

During allocation/execution, MPE/3000 keeps track of each code segment by maintaining information about its nature and current location in the code segment table (CST). Similarly, information about the stack is dynamically recorded in the data segment table (DST).

A particular program can be run by many user processes simultaneously, with all processes accessing the same copy of code. But unlike the program code segments, the data segment containing the stack is private to each user's process and cannot be shared among others. As execution progresses, data enters and leaves the stack dynamically. Within the stack, data is arranged as a linear group of items accessed from one end (called the *top-of-stack*). When the last instruction in a process is executed, MPE/3000 releases those segments associated only with that process, including the stack segment, and deletes their related entries in the CST and DST. (However, shared code segments are not released until the last process using them is deleted.) All descendants of this process are deleted, and all files opened by it are closed. (CST and DST entries assigned to the released segments can be re-assigned.)

PCB/CODE SEGMENT/STACK INTERACTION

Each PCB contains all information needed to control a process. This information includes the priority of the process, and pointers to queues and ancestor and descendent processes.

Each code segment executed by a process can contain one or more program units that include calls to procedures elsewhere in this segment or in another segment. Within a program unit, there are generally many executable machine instructions. Each procedure call also references machine instructions that handle the procedure requested.

When a code segment is executing in main memory, it is defined by pointers in three hardware registers: the Program Base (PB), Program Counter (P), and Program Limit (PL) registers (Figure 2-6). (There is only *one* set of these registers; at any particular instant, their contents refer to the code segment currently in execution.) The PB register contains the absolute address of the starting location of the segment in main memory. The P register holds the absolute address of the instruction currently being executed. The PL register indicates the absolute address of the last location of the code segment. Execution can only be transferred from this segment by an interrupt or by a call or exit instruction referencing a procedure in another segment, in which case the PB, P, and PL registers are re-set to reflect the characteristics of the new segment. Whenever an instruction is executed, the PL and PB registers are checked to ensure that referenced addresses fall within the proper segment boundaries. This *bounds check* guarantees that other programs and the system itself are protected against improper access. Since all addresses within a code segment are relative to the contents of the three program registers, a segment can be relocated anywhere in main memory and only the register contents need be changed to reflect this transfer.

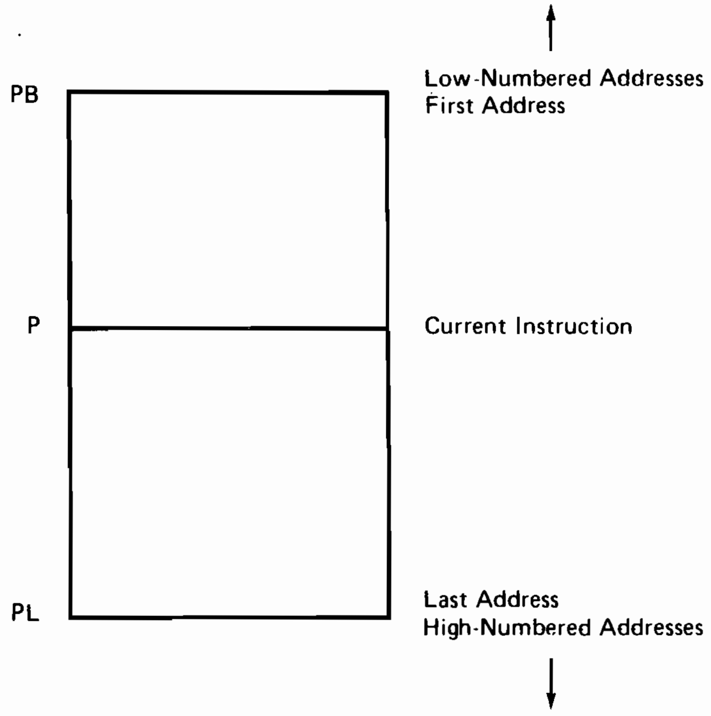


Figure 2-6. Code Segment and Associated Registers

As with code segments, there are normally several stacks resident in memory. (One stack exists for each process.) But since the execution of any process is interleaved with that of others, only one stack is active at any particular instant. Most dynamic computational operations take place on the stack. The last element added to the stack is placed in the word at the *top-of-the-stack*. In this position, it will be the first element removed when the process associated with the stack requests data from the stack. (In other words, the last element in is the first one out.) Each time data is added to the stack, the previous top element becomes the second element from the top; each time the topmost element is removed, the second element returns to the top of the stack.

Programmers operate *directly* on elements in the stack only when using the SPL/3000 language. However, any program in any other language references the stack implicitly when it manipulates data, although the user need not be aware of this.

The stack data segment consists of two general areas: a process control block extension (PCBX) area and a stack area. The *PCBX* contains certain frequently-needed information for process control that need not reside in main memory, such as register settings and file pointers; it is always contiguous with the stack segment. (In general, the PCB is used when the process is *not* running in main memory; the PCBX is used when it is.) The *stack area* is the most significant part of the stack data segment; this is the area where the user's data is stored and manipulated. The boundaries of this area, and its logical subdivisions, are delimited by pointers in registers (Figure 2-7). (At any instant, these registers always apply to the stack belonging to the currently-executing process.)

The Data Limit (DL) register contains the absolute address of the first word of the stack area in main memory. The area between this address and that stored in the Data Base (DB) register is an SPL/3000 *user-managed area* that can be accessed and used only through SPL/3000 programs, such as compilers. These programs sometimes require this space for SPL/3000 own-arrays. (A part of this user-managed area, that between the addresses DB-10 and DB-1, is the subsystem data area reserved for data used during subsystem operation.)

The DB register points to the beginning of the *global area* within the stack area. This area is used for global variables (those declared within the data group of an SPL/3000 main program, and thus usable by any procedure within that program). This area also contains global arrays and pointers to those arrays. The end of the global area is initially indicated by the Stack Marker (Q) register, which also denotes the beginning of the *local area*. (As will be illustrated later, the Q-register pointer changes whenever the running process calls or exits from a procedure.) At any given time, the local area contains data local (relating only) to the procedure currently in execution. The end of the local area is delimited by the Top-of-Stack (S) register, which points to the last item in the stack. All storage between the addresses in the S register and the Stack Limit (Z) register, is unused space that remains available for additional data. The Z register indicates the last main memory location that can be used by user data in the stack area. Beyond the address in the Z register, a special zone is available for stack overflow — a condition that occurs when the S-register pointer must be moved beyond the Z-register pointer and space must be provided for certain end-of-stack information.

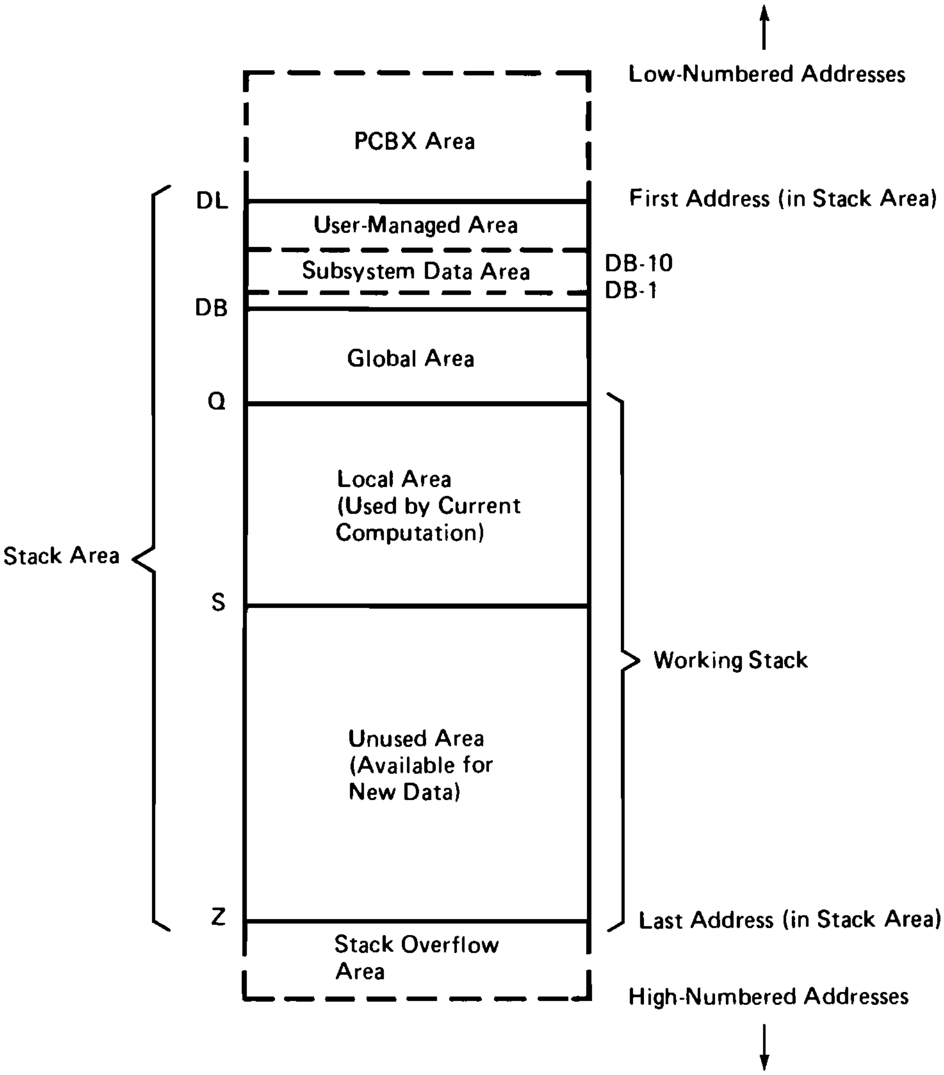


Figure 2-7. Data (Stack) Segment and Associated Registers

The contents of the DL and Z registers delimit the boundaries of the stack area. Through bounds checks that reference these registers, MPE/3000 (and certain hardware provisions) ensure that the user's data remains within these limits, and that no other user accesses this area. Thus, the privacy of the user's stack is guaranteed. All locations in the stack are addressed relative to the DB, Q, or S addresses.

Although the top-of-stack is logically indicated by the S register, up to four of the topmost elements of the stack can actually exist in central processor registers (the TR registers) rather than in main memory — in effect, the stack "spills over" from memory into these registers. These conditions greatly enhance processing speed. The number of TR registers currently in use is indicated by the SR register. The absolute address of the last item of the stack actually residing in main memory is stored in the SM register. The contents of the S register are actually denoted by

$$S = SM + SR$$

Thus, the S register is said to contain the *logical* top-of-stack rather than the actual top-of-stack in memory.

Before a process begins execution, stack space is first reserved for global data, beginning at the DB-address and terminating at the Q-address, which denotes the beginning of the dynamic *working stack* (Figure 2-8A). At this time, no data is stored beyond the Q-address, and so the the S register also points to the same address as the Q register — that is, the bottom and top of the working stack coincide. But, as the process begins execution, data is added to the stack, and the S-pointer (top-of-stack) moves away from the Q-pointer (Figure 2-8B). If, at some point, the process encounters a procedure call, a new area for data local to that procedure must be defined. To do this, the system hardware places a group of four words called the *stack marker* on top of the stack to save information necessary to re-create the currently-defined local area later. The Q and S registers are then pointed to the top word of the stack marker, which also delimits the beginning of a new, fresh and unique local area for the procedure just called (Figure 2-8C).

The words in the stack marker preserve the state of the machine at the time of the procedure call. These words contain the following information, (shown in order ascending toward the Q-address):

Word	Contents
Q-3	Current contents of the Index (X) register.
Q-2	The return address for the *code* segment, denoted by P+1 (relative to the PB register).
Q-1	The current contents of the Status register (which includes the number of the code segment containing the calling procedure).
Q-0	The delta-Q value, which is the number of words between the new and previous Q-locations, enabling the later re-setting of Q to its old value.

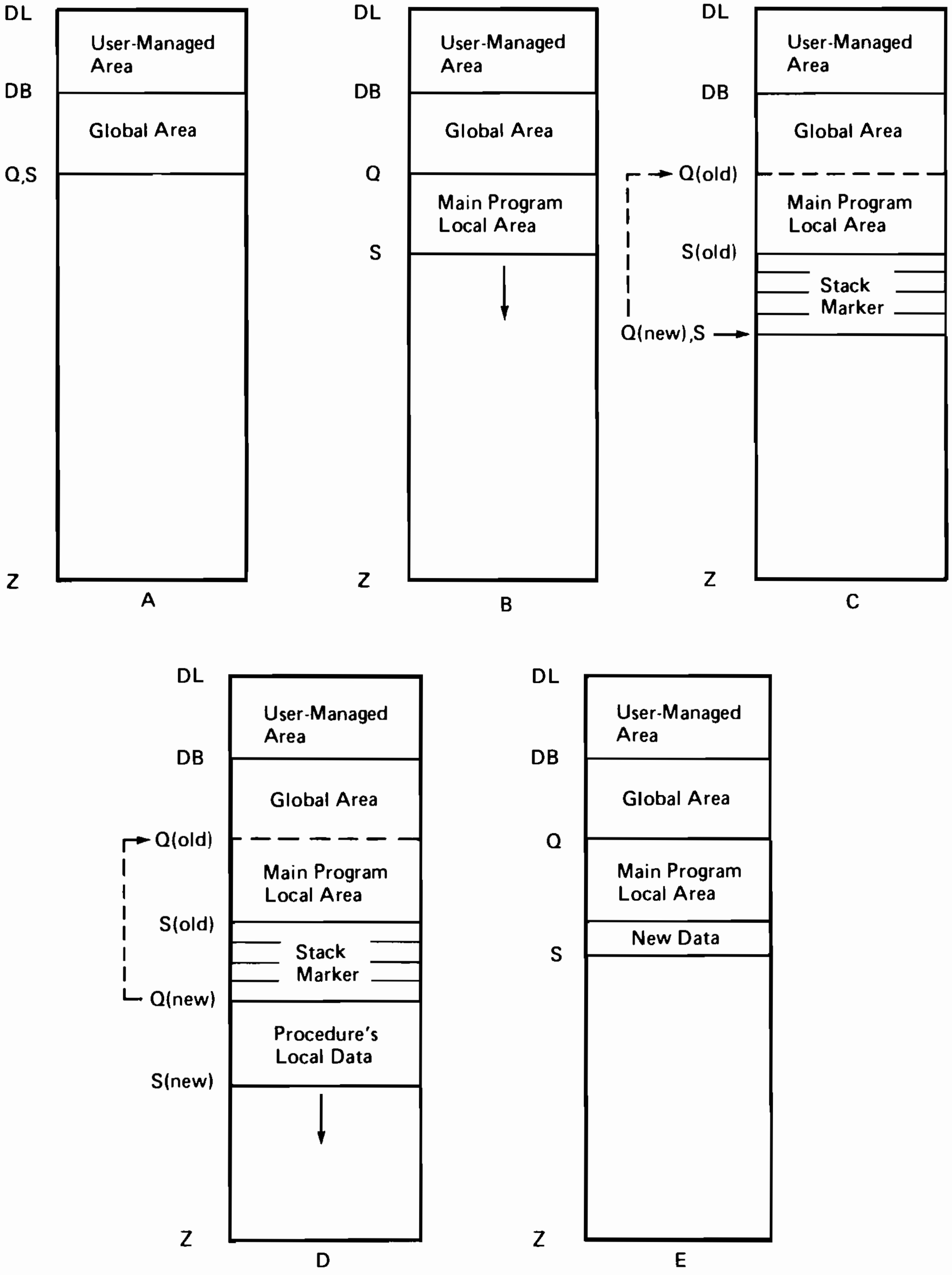


Figure 2-8. Stack Operation

As data is added to the stack during execution of the new procedure, the S-pointer moves away from the new Q-pointer, reflecting the latest data added (Figure 2-8D). When the procedure exits back to the main program, the new local data area is deleted from the stack, the stack marker is used to restore the Q-pointer to its previous setting, any value returned by the procedure is left at the new top of the stack, and the S-pointer is set to indicate the new top-of-stack (Figure 2-8E). This results in a "clean" stack from which temporary data local to the called procedure is eliminated because it is no longer needed.

Whenever a procedure is called, the Q and S registers are manipulated in this manner. The Q register changes with each procedure call and exit; the S register may change when an instruction references data. Thus, when a process executes a main program (outer block) that calls three procedures, there will be a maximum of four local areas (one for the main program and three for the procedures called by it) on the stack. Each procedure's local area will be delimited at its base by its own stack marker. Within these stack markers, the delta-Q words will form a logical chain that links the present Q register setting back to its initial value.

MEMORY MANAGEMENT

The MPE/3000 virtual memory, which frees the user from the restrictions of a physical memory of constant size, consists of main memory and portions of disc. Both the dynamic swapping of segments from disc to main memory, and their dynamic relocation within main memory, are controlled by the Memory Management System through segment descriptors in the CST and DST. When additional memory is required during execution of a process, executed code segments are overlayed by incoming segments rather than actually being swapped out; this saves considerable memory overhead. (If needed later, re-entrant copies of these code segments can be brought in again from disc.) Because the content of a data segment changes continually, however, it is sometimes necessary to swap such a segment both to and from disc to preserve and maintain it until the process completes execution.

Since code can be shared between programs, both the need for multiple copies of programs or routines, and time devoted to swapping between disc and main memory, are reduced.

The user's stack is not the only data area handled as a segment in MPE/3000. In addition, extra data segments (auxiliary to user stacks) and input/output buffers are also managed in this way by the Memory Management System.

Main-Memory Linkages

Main memory is organized into a sequence of variable-length, linked areas that facilitate the insertion or deletion of segments. The memory linkages contain the following information about each area:

- Availability (in use or available)
- Size
- Type (code or data)
- Table pointer (to the CST or DST entry relating to the area)

Those areas that are available are linked together through an Available-Space List; as areas in use are released, they are added to this list. Adjacent available areas are combined to form one contiguous area. Requests for memory can be implicit (through a CST presence-trap) or explicit (through calls from the system or user). To allocate memory, a space-allocation routine interrogates the Available-Space List to find the first area large enough to satisfy the request. If the search succeeds, all or a portion of that area is allocated. If the search fails, one or more overlays is performed to free an area of adequate length.

Assigned storage is linked according to the priority and frequency of its access. Frequently-accessed segments with high priority are located at or near the end of the Assigned-Storage List; infrequently-accessed segments with low priority are located near the head of this list. When an overlay is required, the list is searched, beginning with the head entry. Segments declared main-memory resident, of course, are not overlayed. If a data segment is selected for overlay, and it has been modified in memory, it is copied to a reserved overlay-storage area on disc. (Code segments are overlayed without being copied, since copies already exist on disc.) Each process is assigned such a reserved area when it is initialized; it may contain either code or data segments.

Main-Memory Use

When MPE/3000 is initialized, part of main-memory (usually in the low-address area) is pre-allocated to MPE/3000 or to the hardware for resident routines and tables; this area is not available for swapping. The remaining portion of main memory is established as the linked areas described above. During MPE/3000 initialization, some of this space is allocated for the CST, DST, and PCB tables, and the remainder is available for overlays (and so described in the Available-Space List).

USER JOB PROCESSING

The following paragraphs discuss how MPE/3000 concepts interrelate, from the standpoint of the user's program.

Batch Programs (Jobs)

To illustrate how a typical batch program (a job) is processed, consider a small FORTRAN source program punched on cards. The user arranges the deck so that the MPE/3000 commands, the FORTRAN program, and any input data are presented in the proper order, and submits this deck to the computer operator to run.

The operator enters the job through a card reader. MPE/3000 assigns the job a unique system job number, and schedules it for access to the central processor. When the central processor is available, a special system process determines whether the attempted access is valid (by checking the user's identity and passwords) and requests the user controller process (UCOP) to create a job main process (JMP) that uses Command Interpreter code to run the job (Figure 2-9A, B). The JMP completes the job initialization and becomes the process responsible for handling interactions (commands) between the user and MPE/3000 (Figure 2-9B).

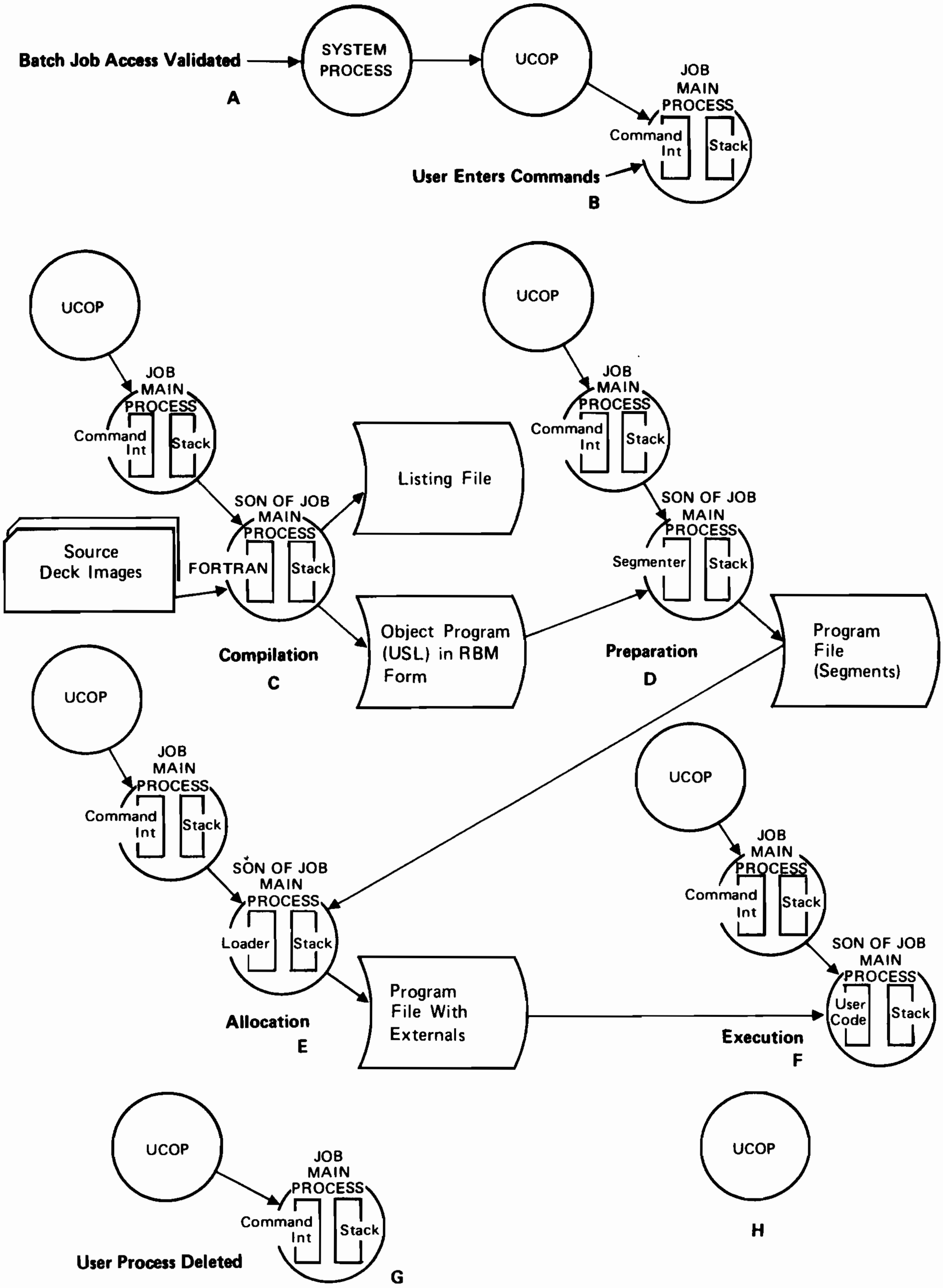


Figure 2-9. Batch Job History

When this process encounters a user request to compile, prepare, and execute the user's program, it creates another process that calls the FORTRAN/3000 compiler into virtual memory and executes the compiler code. (Figure 2-9C). This process translates the FORTRAN program from source to object code and organizes this code into RBM's stored as a USL on disc. In this process, the compiler interacts with the Input/Output and File Management Systems to read input and write output. Next, the main process creates a process that runs the segmenter subsystem, which reads the program from the USL and prepares it for execution by arranging it as a set of linked segments on a program file (Figure 2-9D). Then, the loader allocates the segments in virtual memory on disc, and satisfies external references to library routines (Figure 2-9E). The initial stack area is also defined at this time. Then, the code segment containing the entry point of the user's program, and the stack segment, are moved into main memory and execution of the user's process begins. (Figure 2-9F). Files needed by the job, and their associated devices are assigned as requested.

As processes are created by and for the job, they are entered into queues and scheduled for access to the central processor. Each process may create many descendent processes as the job progresses. Various processes share the central processor through multiprogramming, and in this way are run concurrently. The code segments belonging to each process obtain and operate on data from the stack associated with that process. As each process proceeds, segments are swapped into main memory, as they are needed, by the Memory Management System, which calls the Input/Output System to handle this function. Data enters and leaves the stack area dynamically, handled by the File Management System. Program output is transmitted to a line printer and printed by an input/output process as the job is run.

As the job progresses, the operator receives any pertinent messages. He can terminate the job at any time. When any process (including the main process) is completed, it and its descendants are deleted from the system; all files opened by this process are closed (Figure 2-9G, H). As the job is completed, system information related to the job is printed, including the time and date, and central processor time used.

Interactive Programs (Sessions)

As an example of how an on-line, interactive program is handled, consider a BASIC-language program run by a user at a remote terminal.

The user contacts the system by turning the terminal on (if the terminal is connected directly) or by dialing the system (if the terminal is connected by switched telephone lines). In response, a special system process determines the validity of the access and requests the UCOP to create a session main process (SMP) that uses the Command Interpreter code. The SMP completes the session initialization and becomes the process responsible for MPE/3000 interactions (commands) between the user and the system. The user next enters a command to access the BASIC/3000 Interpreter. A process is now created that handles input in the BASIC language. (Only one copy of the BASIC/3000 Interpreter is needed in main memory no matter how many users are programming simultaneously in BASIC; however, each user accesses the interpreter through a separate process.)

Many independent user processes share the central processor through time-slicing. Some commands create processes that may, in turn, create descendant processes. When required by each process, data segments are swapped to and from main memory. The Memory Management System interacts with the Input/Output System to accomplish this. The File Management System is invoked to allocate devices for files whenever the files are opened. The user can direct output to the terminal, or he can have it transmitted to a printer.

The user can terminate the session at any time. When he does this, the SMP prints the time and date, central processor time, and terminal connect time.

ACCOUNT/GROUP/USER ORGANIZATION

When a user logs on to MPE/3000, two basic elements must be defined: an identifiable unit to which system resources (such as disc file space and central processor time) are allocated and charged, and a local set (domain) of disc files accessible by the user. The basic unit to which resources are assigned is the *account*; this is the major "billable unit" in MPE/3000. Associated with each account is a unique file domain, a set of *users* who can access MPE/3000 through this account, and a set of *groups* which partitions the account's accumulated resources and divides its file domain into private sub-domains.

Each account is defined, modified, and deleted by commands issued by a user with the MPE/3000 System Manager Capability, who has ultimate control over access to the system and allocation of its resources. Each account is identified by a name. Optionally, a password can be associated with the account to validate a user's ability to access MPE/3000 under this account at log-on time. A maximum priority also is associated with the account; this designates the highest priority at which any process run under this account can be scheduled.

Limits are assigned for maximum disc file space, central processor time, and on-line connect-time permitted each account; running counts of the use of these resources are maintained for billing purposes. To maintain an account, the user acting as System Manager grants a user the Account-Manager Capability. This account-managing user may in turn, assign the same capability to other users in his account.

The users and groups associated with each account are defined by commands issued by the account-managing user. Each user is identified by a name (unique to this account) and optional log-on password. He is assigned a maximum allowable priority for his processes, which cannot exceed the priority allowed to his account. Each account possesses a public group (to which all of its users have read and program-execution access) in addition to other groups that may be covered by various security provisions. Each group is identified by a name unique within its account, and optionally, by a password used to validate access to the group and its files at log-on time.

As with an account, limits are assigned for the maximum disc file space, central processor time and on-line connect time usable by a group; and running counts of resources used by the group are maintained. (File space is always charged to the group containing the file, rather than the group to which the user who created the file was logged on.)

Any MPE/3000 installation can contain several accounts; each account can have several users and groups associated with it; each group can possess several files (see Figure 2-10) which constitute a subset of the file domain. When the user logs on, he specifies the account, user, and group names (and, if required, the account, user, and group passwords). Furthermore, any file in a group may also be protected by a *lockword* required at any time the user accesses the file during the course of his job or session (in addition to standard file security mechanisms described later.)

Each user can be associated with a *home group* by the user managing his account. If the user does not specify a group when he logs on, he is given the home group by default.

Once the standard user has established communication with MPE/3000, if the normal (default) system security provisions are in force, the user has unlimited access to all files in his log-on group and home group. Furthermore, he can read, and execute programs residing in, files in the public group of his account or in the public group of the system account. He cannot, however, access other files in the system in any way.

The normal MPE/3000 security provisions can be overridden at the account, group, or file level, (by System Manager, Account Manager, or standard users, respectively) to provide more or less restriction to users. Furthermore, users with special capabilities (discussed next in this section) are generally subject to fewer restrictions.

CAPABILITY SETS

The HP 3000 Computer is used by a large variety of programmers, ranging from those who want to run simple applications programs in BASIC to system programmers who are actually modifying MPE/3000. To protect the system and its users in general, users with System and Account Manager Capabilities can limit access to special system capabilities only to those who fully understand their correct use. This is done through capability sets. Specifically, when a System-Manager User creates an account, he defines for it a capability set that determines whether or not users communicating with MPE/3000 through this account can be allowed certain functions. When an Account-Manager User defines the users of his account, he associates with each user an individual capability set that may allow the user some or all of the general account capabilities. Each capability set contains three types of attributes: user, file-access, and capability-class. A fourth attribute, the local attribute, may also be defined. The combination of these attributes determines the set of commands and intrinsics available to the user. This division of commands and intrinsics greatly simplifies use of the system from the standpoint of each individual user — it defines the extent to which he must understand and interrelate with MPE/3000, and permits a user to ignore aspects of MPE/3000 that do not apply to him.

Capability sets are also defined for groups by the Account-Manager User. Group capability sets contain only one type of attribute — the capability-class attributes. The capability set for a group may allow that group some or all of the capability-class attributes defined for the account to which the group belongs. The group capabilities relate to the user's capabilities as noted at the end of this section under *Program Capability Sets.*

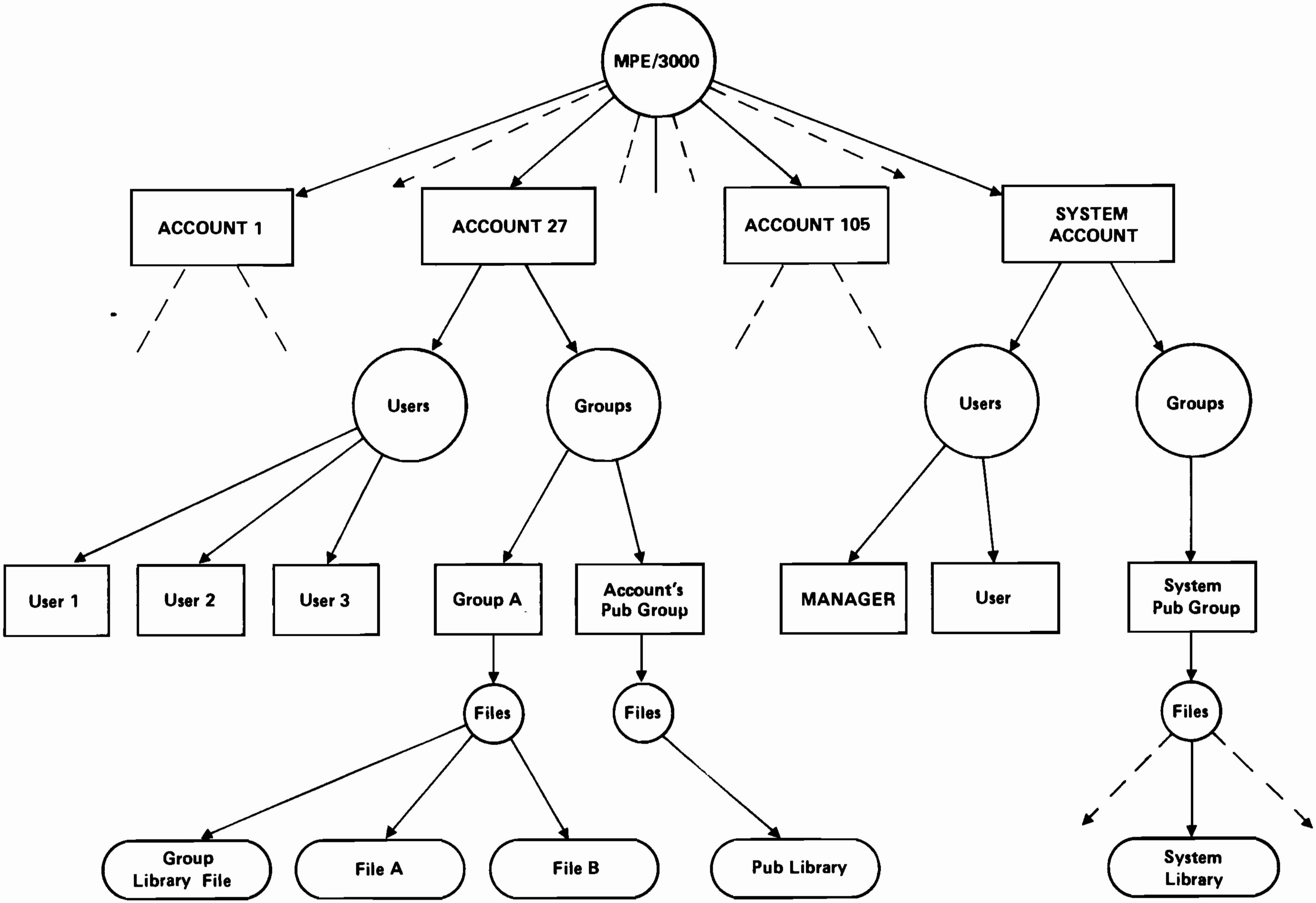


Figure 2-10. Account/User/Group Organization

As noted at the end of this section; capability-class attributes are also associated with each program on a program file, passed as parameters (in the command that prepares the program) to the MPE/3000 Segmenter.

User Attributes

The user attributes designate the general level at which the user interfaces with MPE/3000. These attributes can be assigned in any combination, and define capabilities *in addition* to those of a standard user.

SYSTEM MANAGER ATTRIBUTE. Grants the user the capability to manage the overall system and create the accounts within it. The first user with the System Manager Attribute is designated on the system tape furnished the customer. He, in turn, can designate other users having the same or different capabilities.

ACCOUNT MANAGER ATTRIBUTE. Allows the user to manage all users and groups within an account. The first manager for each account is designated by a user with the System Manager Attribute when the account is created. A user with the Account-Manager Attribute, in turn, can assign this attribute to other users in his account.

SYSTEM SUPERVISOR ATTRIBUTE. Allows the user to have day-to-day external control of this system. It allows him to manage scheduling subqueues, alter the system configuration, maintain the system logging facility, and display various items of system information. This attribute may be assigned by a user with the System Manager Attribute.

ACCOUNT LIBRARIAN ATTRIBUTE. Can be assigned to grant a user special file-access modes for maintenance of files within his account. For example, an Account Librarian Attribute may be used to designate users who can purge (but not create or alter) files within the account. (File-access modes such as read-access or write-access, are discussed in Appendix I.) This attribute is assigned by users with the Account Manager Attribute.

GROUP LIBRARIAN ATTRIBUTE. Similar to the Account Librarian Attribute, but limits the special file-access modes allowed the user to his home group. This attribute is assigned by users with the Account Manager Attribute. It could be used, for example, where it is desired that only one user can have the capability to alter files within a particular group. This user could be assigned the Group Librarian Attribute and his access modes could be made greater than those of other users.

DIAGNOSTICIAN ATTRIBUTE. Permits the user to run diagnostic programs for on-line check-out of hardware under the System Diagnostic Monitor (SDM/3000). This attribute is assigned by users with the Account Manager Attribute.

File Access Attributes

The file-access attributes determine whether the user has the capability to

- Save user files permanently
- Access card-readers, line printers, and other non-sharable input/output devices (other than the standard job/session input and listing devices which are available to all users)

These attributes can be assigned in any combination.

Capability-Class Attributes

These attributes define the general MPE/3000 resources available to a user. They allow the user to

- Access MPE/3000 in an interactive on-line mode
- Access MPE/3000 in local, batch processing mode

Most users have *only* these MPE/3000 Standard Capabilities; all users except console operators must have one of these capabilities *at least.*

- Create, handle, and delete processes directly
- Manage data segments (by creating extra data segments)
- Have exclusive use of more than one system resource simultaneously
- Operate in privileged mode (which permits a user to access *all* MPE/3000 resources, including uncallable intrinsics)

MPE/3000 Optional Capabilities

Either the Interactive or Batch-Access Attribute is required to communicate with MPE/3000; most users (including users with System Manager and System Supervisor Capabilities) are assigned both as *standard capabilities.* The remaining capability-class attributes are *optional capabilities.* They are independent and can be assigned in any combination. The optional capabilities are discussed in *HP 3000 Multiprogramming Executive Operating System (03000-90005).* Programmers should be aware that the more powerful the optional capability, the more hazardous its misuse is to the system. Thus, they should use optional capabilities with caution.

Local Attributes

The local attribute is a two-word quantity used only for special applications that require further unique classification of users. MPE/3000 does not reference or make use of this attribute in any way; rather, it is defined arbitrarily by System or Account Managers, and used by accounts or groups for whatever purpose they require.

Program Capability Sets

Capability sets are always associated with prepared programs as well as users.

Each time someone runs a program, MPE/3000 automatically assigns that program the user and file-access attributes of that user. But the capability-class attributes assigned to the program are designated by the user who originally prepares the program; they are passed to the MPE/3000 Segmenter as parameters of the command that prepares the program. If the preparing user does not designate capability-class attributes for his program, MPE/3000 assigns, by default, the standard capabilities possessed by that user — interactive access, batch access, or both. (When programs prepared from passed files or job temporary files are run, they are assigned the standard capabilities (interactive and/or batch access) possessed by the user who runs them.)

If the program resides on a permanent file, the program's capability-class attributes should not exceed those defined for the *group* to which the program file belongs. If they do, the user will not be able to run the program when he attempts to do so.

Because the capability set is associated with the entire set of code segments being run (and hence with the process running them), all procedures, subprograms, and subroutines on those code segments have the same capability. For the same reason, a *user* need not have the same capabilities as the programs he runs.

SECTION III
Communicating with MPE/3000

To communicate with MPE/3000, the user issues commands and intrinsic calls.

Commands are requests issued to MPE/3000 to perform various broad functions external to the user's program. For example, they are used to initiate and terminate jobs and sessions, create and maintain files, compile and execute programs, call various utility subsystems and obtain job status information. A user with System Manager Capability can use them to create or change accounts; a user with System Supervisor Capability uses them to display information, or manage the scheduling queue or System Logging Facility. Commands can be entered through any standard input device, typically the card reader (for jobs) or the terminal (for sessions). Each command is accepted by the MPE/3000 Command Interpreter, which passes it to the appropriate procedure for execution. Following this execution, control returns to the Command Interpreter.

Intrinsic calls are used to invoke MPE/3000 functions requested within a user's program, such as reading, writing on, and updating files, skipping forward and backward on files, or returning system table information to the user's program. Two intrinsics, CREATE (for creating a process) and GETPRIORITY (for rescheduling a process) allow a programmer who has System Supervisor Capability to specify certain parameter values that cannot be specified by other users calling these intrinsics. For this reason, CREATE and GETPRIORITY are discussed in this manual. All other intrinsics are described in *HP 3000 Multiprogramming Executive Operating System (03000-90005).* In an SPL/3000 program, the user writes the intrinsic calls explicitly. In FORTRAN/3000, BASIC/3000, or COBOL/3000 programs, for most applications, the compiler generates any necessary intrinsic calls automatically—they are invisible to the user. At their option, however, FORTRAN/3000, COBOL/3000, and BASIC/3000 users can also directly call intrinsics as their needs require, providing added power and flexibility to these standard programming languages.

The programmer can use intrinsic calls to invoke the Command Interpreter from within his program, and pass to it command images that will be interpreted and executed as the corresponding system commands.

The commands and intrinsic calls discussed in this manual relate to initiating and terminating communication with MPE/3000 and requesting those functions available only to the users with the System Manager or System Supervisor Capabilities. These users also have access to all MPE/3000 standard capabilities, which are discussed in *HP 3000 Multiprogramming Executive Operating System (03000-90005)*; these capabilities include compiling, preparing, and running programs, accessing files, managing libraries and various utility functions.

COMMANDS

Each command entered by the user, whether in a batch job or interactive session, consists of

- A colon (used as a command identifier)
- A command name
- A parameter list (in most cases)

The end of each command is delimited by the end of the record on which it appears—for example, a *carriage return* for terminal input or the end of the card on which it is punched for card input. But, if the last non-blank character of the record is a *continuation character*, as defined later in this section, the end-of-record does not terminate the command. (Users running programs in batch mode should bear in mind that all 80 columns on each card image are scanned by MPE/3000, and thus no characters are ignored.)

The *colon* identifies a statement as an MPE/3000 command. In a batch job, the user begins each MPE/3000 command with this colon in column 1 of the source card (or card image). In an interactive session, however, MPE/3000 prints the colon on the terminal whenever it is ready to accept a command; the user responds by entering the command after the colon. (Interactive subsystems of MPE/3000 also use unique prompt characters; in a session, the prompt character output tells the user that a subsystem is ready.)

The *command name* requests a specific operation, and appears immediately after the colon. Imbedded blanks are not permitted within the command name. The end of the command name is delimited by any non-alphabetic character, normally a blank.

The *parameter list* contains one or more parameters that denote operands for the command. It is required in some commands but optional or prohibited in others. Parameter lists can include positional parameters and/or keyword parameter groups. Within the parameter list, delimiters (commas, semicolons, equal signs, or other punctuation marks) are used to separate parameters or parameter groups from each other, as described below.

Normally, the parameter list itself is separated from the command name by one or more blanks. However, when the first optional parameter in a positional list (as defined below) is omitted, the command name can be followed immediately by any other delimiter. (At the user's option, he can include one or more blanks between the command name and this delimiter.) Any delimiter can be optionally surrounded by any number of blanks, permitting a free and flexible command format.

EXAMPLE:

The following command (RUN) is preceded by a colon and includes a parameter list with two parameters:

```
:RUN PROG, ENTRYX
```

Both decimal and octal numbers are permitted as command parameters. Octal numbers, however, are always preceded by a percent sign (%).

Positional Parameters

With positional parameters, the meaning of the parameter is designated by its position in the list. For example, in the MPE/3000 command to compile a FORTRAN/3000 program, the parameter specifying an input file always precedes the one that specifies an output file. Positional parameters are mutually-separated by commas or semi-colons. The omission of an optional positional parameter from within a list is indicated by two adjacent delimiters; the omission of a positional parameter that would otherwise immediately follow a command name is indicated by a comma or semi-colon as the first character in the parameter list. When parameters are omitted from the end of a list, however, no adjacent delimiters need be included to signify this — the terminating *return* or end-of-card is sufficient.

EXAMPLE:

The first command below has its first parameter omitted; the second command has its second (embedded) parameter omitted; the third command has its last two (trailing) parameters omitted; the fourth command has all parameters omitted. (In the second and third commands, the asterisk (*) is not a delimiter, but a special character used to denote a back-reference to a previously-defined file, as described in HP 3000 Multiprogramming Executive Operating System (03000-90005). In each case, the asterisk is considered part of the following parameter.)

```
:FORTRAN ,USFL,LISTFL,MFL,NFL
:FORTRAN *SOURCEFL,,LISTFL,MFL,NFL
:FORTRAN SOURCEFL,USFL,*LISTFL
:FORTRAN
```

A further example illustrates the relationship between the positions of parameters in a list and their meanings:

EXAMPLE:

In the following command (:FORTRAN), three positional parameters appear: INP refers to an input source file, OUT indicates an object (USL) output file, and LST indicates the listing output file. For the :FORTRAN command, these three fields always have the same meanings. (Note that the second delimiting comma in the parameter list is followed by an optional blank. In future examples, for clarity, delimiters will always be followed by blanks.)

```
:FORTRAN INP,OUT, LST
```

Keyword Parameters

When a parameter list is so long that use of positional parameters becomes difficult, keyword parameter groups are often used. The meaning of such a group is independent of its position in the list. A keyword parameter group is designated by a keyword that denotes its meaning, and *optionally* an equal sign and one or more sub-parameters. (Each keyword group is preceded by a semi-colon. When more than one sub-parameter appears in a group, they are usually separated from each other by commas. Like other delimiters, semi-colons and commas can be optionally preceded or followed by blanks.) With respect to each other, keyword groups can appear in any order. When keyword groups and positional parameters both appear in a list, however, the positional parameters always precede the keyword groups; when this occurs, and *trailing* parameters are omitted from the positional group, their omission need *not* be noted by adjacent delimiters; instead, the occurrence of the first keyword indicates this omission.

EXAMPLE:

In this example, DL and LIB designate keyword parameter groups. The value 500 and the letter G are subparameters of these groups.

```
:RUN PNAME, ENPT; DL=500; LIB=G
```

Continuation Characters

When the length of a command exceeds one record (source card or entry-line), the user enters an ampersand (&) as the last non-blank character of this record and continues the command on the next record. In this case, the next record must begin with a colon (entered by the user in batch processing, but prompted by the terminal in interactive processing). Optionally, blanks can be embedded between the colon that begins a continuation record, and the rest of the information on that record. Commands can be continued up to 255 characters, including prompting colons and continuation ampersands.

EXAMPLE:

The following command image contains a continuation character at the end of the first line:

```
:RUN PROGB; NOPRIV; LMAP; STACK=500; PARM=5; &
: DL=600; LIB=G
```

In continuing a command onto another line, the user cannot divide a primary command element (a command name, keyword, positional parameter, or keyword sub-parameter) — no primary element is allowed to span more than one line.

MPE/3000 does not begin interpretation of a command until the last record of the command is read. For interpretation, all records within the command are concatenated, and all prompt characters and continuation ampersands are replaced by one or two blanks.

Command Description Format

To help clarify the command descriptions that appear throughout this manual, system output is *underlined.* Input information is not underlined. (In cases where differences in output occur between batch and interactive processing, *interactive* output is assumed unless otherwise noted.) For both input and output, literal information that always appears exactly as shown is designated by *CAPITAL LETTERS AND SPECIAL CHARACTERS IN ITALICS;* on the other hand, symbols that represent variable information are indicated by *lower-case italics.*

Optional information is indicated by surrounding [*brackets*]. (However, the user does not enter these brackets as part of the command.) {*Braces*} indicate that one of the items included is required and must be entered by the user.

EXAMPLE:

The colon is output by the terminal during interactive sessions (as shown by the underline); SHOWQ is a literal command name entered by the user, (as indicated by capital letters); and subqueuename is a variable parameter input by the user (as shown by lower-case letters). The subqueuename parameter is optional (as indicated by brackets).

```
:SHOWQ [subqueuename]
```

When more than one item is enclosed vertically in brackets, this indicates that exactly one of these items *may* be specified.

EXAMPLE:

```
:LISTACCT [@       ] [,listfile]
          [acctname]
```

When more than one item is enclosed vertically in braces, this means one item *must* be specified.

EXAMPLE:

```
:SAVE       {$OLDPASS,newfilereference}
            {tempfilereference        }
```

If items are shown within vertical and disjoint brackets, this signifies that they are all optional and that those specified by the user can appear in any order.

EXAMPLE:

```
:RUN progfile [,entrypoint]
     [;NOPRIV]
     [;LMAP]
     [;MAXDATA = segsize]
         .
         .
         .
```

Command Errors

All MPE/3000 commands issued, and any system messages for the user, are copied to the job/session listing device.

When MPE/3000 detects an error in a command, it suppresses execution of that command and prints an error message. If this occurs during a batch job, (and no :CONTINUE command precedes the erroneous command, as discussed later) all information between the erroneous command and the end of the job is ignored, and the job is aborted. If an error occurs during an interactive session, an indication is printed and control returns to the user; he may then re-enter this command correctly, or enter any other command he desires simply by pressing the carriage-return key and entering the command. Alternatively, if he desires a fuller explanation of the error, he can request it by entering any character directly after the error-number shown in the error message. In response, the explanation appears on the next line. Error messages and on-line error recovery are discussed in *HP 3000 Multiprogramming Executive Operating System (03000-90005).*

INTRINSIC CALLS

A major advantage of MPE/3000 intrinsics is that they can be called directly from programs written in standard, higher-level programming languages (FORTRAN/3000, COBOL/3000, and BASIC/3000) as well as from SPL/3000 programs. The rules for invoking intrinsics from programs written in standard languages are presented in the manuals covering those languages. However, because intrinsic calls are issued primarily by users programming in SPL/3000, the rules for issuing these calls from SPL/3000 programs are summarized below.

Before an intrinsic can be called from an SPL/3000 program, it must be declared within the declaration portion of the program, following all data declarations, like any other SPL/3000 procedure. This can be done by writing the standard PROCEDURE declaration. Normally, such a declaration would contain both the procedure head and the procedure body (code). But, since an intrinsic is an *external* procedure, the user follows the normal SPL/3000 conventions for such procedures by writing only the intrinsic *head* and including within it the OPTION EXTERNAL notation. (EXTERNAL signifies that the body code will be supplied from a library and linked to the user's program during program allocation.) In this manual, the complete head format for each intrinsic is presented with the discussion of that intrinsic. (Note that in SPL/3000, each statement is delimited by semi-colons.)

EXAMPLE:

The format for the head of the GETPRIORITY intrinsic is presented in Section VIII. Using this format, the user could write a declaration that will enable him to call the GETPRIORITY intrinsic later in his program. The declaration could appear as follows. Note that the user must include the OPTION EXTERNAL notation.

```
PROCEDURE GETPRIORITY (FN,PC,RNK);
VALUE PC,RNK;
LOGICAL PC;
INTEGER FN,RNK;
OPTION VARIABLE,EXTERNAL;
```

Because some intrinsic heads are rather long, the SPL/3000 programmer can save time and effort by using an alternative method of declaring intrinsics; the INTRINSIC declaration. This is written in the following format

INTRINSIC intrinsicname, intrinsicname, . . . , intrinsicname;

In the *intrinsicname* list, the user names all intrinsics he intends to call within his program. (When more than one is named, the names are separated by commas.) He then need not write the heads for these intrinsics.

EXAMPLE:

To use the INTRINSIC declaration to declare the FOPEN, FREAD, FWRITE, and FCLOSE intrinsics, the user could write:

```
INTRINSIC FOPEN,FREAD,FWRITE,FCLOSE;
```

Regardless of whether the programmer declares an intrinsic with a PROCEDURE declaration or an INTRINSIC declaration, he must know the formal head format for each intrinsic, since this tells him the number and type of parameters that are used in calling that intrinsic.

The user calls an intrinsic in his program exactly as he calls any SPL/3000 procedure: he writes the intrinsic name and a parameter list, enclosed in parentheses. These elements follow the positional format shown in the intrinsic head; parameters are separated from each other by commas.

EXAMPLE:

A call to the GETPRIORITY intrinsic, shown earlier, could be written as

```
GETPRIORITY (FWORD,PRWORD,RVAL);
```

If the OPTION VARIABLE notation appears in the intrinsic head, some of the intrinsic parameters are optional. In this manual, these optional parameters are indicated as **bold** face in the intrinsic head formats.

Since all intrinsic parameters are positional, the user indicates a missing parameter within a list by following the previous delimiting comma with another comma by itself.

EXAMPLE:

The second parameter is missing:

```
CREATE (PNAME,,PROG);
```

If the first parameter is omitted from a list, this is indicated by following the left parenthesis with a comma. If one or more parameters are omitted from the end of a list, this is indicated by simply writing the terminating right parenthesis after the last parameter included.

NOTE: In some intrinsic calls, input parameters are passed to the intrinsic as words whose individual bits or fields of bits signify certain functions or options. In cases where some of the bits within a word are described in this manual as "not used by MPE/3000," the user is advised to set such bits to zero. This will help ensure the compatibility of his current programs with future releases of MPE/3000.

In cases where output parameters are passed by an intrinsic to words referenced by a users' program, bits within such words that are described as "not used by MPE/3000" are set as noted in the discussion of the particular parameter.

Intrinsic Description Format

In this manual, the complete intrinsic declaration head is shown with the discussion of each intrinsic; included within this is the intrinsic call format, distinguished from the remainder of the head by a box. This format appears as follows:

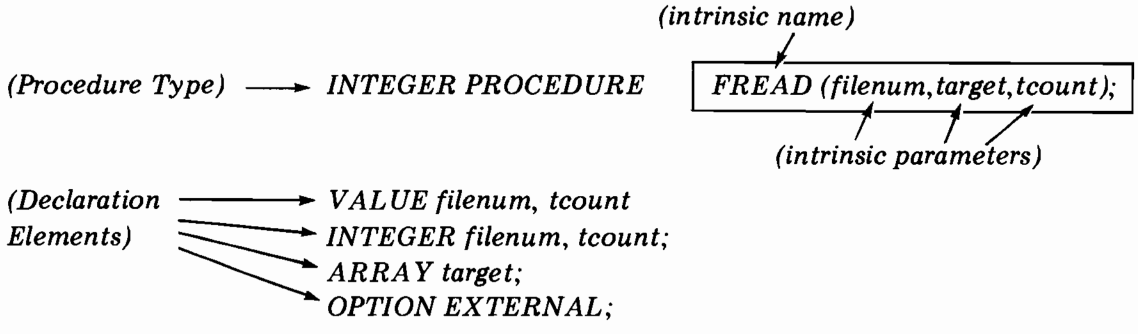
procedure type PROCEDURE | *intrinsic name (parameter list)* |
declaration elements

The *procedure type* applies only to *type intrinsics*—those that return a value to the user's program in response to the intrinsic call. This describes the type of data returned, such as integer, real, or byte values. The data is returned through the intrinsic name, as demonstrated in later examples. Additionally, type intrinsics can be directly related to functions called in FORTRAN/3000 programs, as noted in the manual *HP 3000 FORTRAN (03000-90007).*

The *declaration elements* describe each parameter (variable, pointer, label, and array) in the parameter list, and its characteristics. If the VALUE notation is present in the declaration, either a numeric value (expression) or a symbolic identifier may be specified for the parameters indicated by this notation. If VALUE is not present, only symbolic identifiers may be used for parameters.

EXAMPLE:

This is the intrinsic description format for the FREAD intrinsic:



```
INTEGER PROCEDURE FREAD (filenum,target,tcount);
VALUE filenum, tcount
INTEGER filenum, tcount;
ARRAY target;
OPTION EXTERNAL;
```

Intrinsic Call Errors

Some intrinsics alter the *condition code*, made available to the user's FORTRAN/3000 and SPL/3000 programs through the Status register. Since the contents of this register change continually, the user should check this register for condition codes immediately upon return from an intrinsic. A condition code is always one of the following, and has the general meaning indicated. The specific meaning, of course, depends upon the intrinsic called.

Condition Code	General Meaning
CCE	Condition code is zero. This generally indicates that the user's request was granted.
CCG	Condition code is greater than zero. A special condition occurred but may not have affected the execution of the user's request. (For example, the request was executed, but default values were assumed as intrinsic call parameters.)
CCL	Condition code is less than zero. The user's request was *not* granted, but the error condition may be recoverable. Beyond this condition code, some intrinsics return further error information to the user's program through their return values.

Two types of errors may be encountered when an intrinsic is executed. The first, denoted by the CCG or CCL condition codes, is generally recoverable and is known as a *condition-code error*. The second type is an *abort error*, which occurs when a calling program passes illegal parameters to an intrinsic, or does not have the capability class demanded by the intrinsic. The user's program may recover from this error or take some other appropriate action if an appropriate error-trap procedure has been defined (as discussed in *HP 3000 Multiprogramming Executive Operating System*). Otherwise, his program terminates, and an abort-error message appears on his output device. If the program was entered in a batch job, MPE/3000 removes the job from the system (unless a :CONTINUE command, defined later in this section, precedes the error); if it was entered in an interactive session, MPE/3000 returns control to the user at the terminal. Abort-error messages are described in *HP 3000 Multiprogramming Executive Operating System.*

NOTE: Whenever a callable intrinsic is invoked by a process running in either the non-privileged mode, or the privileged mode with the DB register pointing to the DB area in the user's stack, a bounds check takes place to ensure that all parameters in the intrinsic call reference addresses that lie between the DL and S addresses in the stack (prior to the intrinsic call); if an address outside of these boundaries is referenced, an abort-error occurs.

When a callable intrinsic is invoked by a process running in the privileged mode, and the DB register points to a data segment other than the user's stack segment, the results depend on the particular intrinsic. Most intrinsics immediately abort in this case. Others (indicated in Appendix C) are allowed to execute following a *bounds-check that ensures that all parameters in the intrinsic call reference addresses that lie within the data segment; any boundary-violation results in an abort-error.*

BATCH JOBS

The user can initiate and terminate batch jobs through any device that accepts serial input (such as a card reader, tape unit, or terminal used non-interactively) located at the computer site, or through a terminal used non-interactively and located remotely, as discussed below.

Initiating Batch Jobs

A batch job consists of a set of cards (or card images) that begins with a :JOB command, contains additional MPE/3000 commands, user programs, and optional data, and terminates with an :EOJ command. The :JOB command initiates the job by establishing contact with MPE/3000. If it accepts this command, MPE/3000 begins job processing. When processing begins, commands are sequentially accepted from the job until an :EOJ command is encountered or until the job is halted by one of the abnormal events described under "Premature Job or Session Termination."

A job can be entered through any device configured to accept jobs. The user (or computer operator) first requests an interrupt on the job input device as follows:

1. For card readers or magnetic tape units, the user turns the device on-line.

2. For terminals, the user turns the terminal on, dials MPE/3000 (if the terminal is connected over switched telephone lines), and presses the *return* key (to generate a carriage return).

NOTE: *The device through which a job/session is entered is called the job/session input device, and is recognized as the standard input file for the job/session. Each potential job/session input device is related, during system configuration, to a corresponding job/session listing device that is recognized as the standard output file for the job/session (unless another device is designated by the user for this purpose). There is one job/session input device and one corresponding job/session listing device for each job/session.*

The user writes the :JOB command in the following format. (The keyword parameter groups, of course, can be specified in any order.)

```
:JOB  [jobname,] username[/upass].acctname[/apass] [,groupname[/gpass]]
      [;TERM=termtype]
      [;TIME=cpulimit]
      [;PRI=executionpriority]
      [;INPRI=selectionpriority]
      [;OUTCLASS=outputclass]
```

The parameters have the meanings noted below.

> NOTE: *In MPE/3000, names (such as the jobname, username, upass, acctname, apass, groupname, and gpass parameters noted below) consist of from one to eight alphanumeric characters, beginning with a letter, unless otherwise specifically indicated. Embedded blanks are not permitted within username.accountname specifications.*

jobname — An arbitrary name used in conjunction with the *username* and *acctname* parameters to reference this job in other commands. If omitted, a *null jobname* is assigned. (Optional parameter.)

> NOTE: *A fully-qualified jobname consists of [jobname,] username.acctname.*

username — The user's name, as established in MPE/3000 by the user with Account Manager Capability. This name is unique within the account. (Required parameter.)

upass — The user's password. (Required if the user has been assigned a password by the user with Account Manager Capability.)

acctname — The name of the user's account, as established by the user with System Manager Capability. Note that this parameter is preceded by a *period* as a delimiter. (Required parameter.)

apass — The account password. (Required if the account has been assigned a password by the user with System Manager Capability.)

groupname — The name of the group (established by the user with Account Manager Capability) that the user wishes to use, during his job, for his local file domain and for central processor time accumulation charges. If omitted, MPE/3000 assigns the user's home group. (Optional parameter for users with a home group; Required parameter for users without a home group.)

gpass The group password. When a user logs on under his home group, no password is needed. (Required parameter for some groups.)

termtype The type of terminal used for input.

(MPE/3000 uses the *termtype* parameter to determine device-dependent characteristics such as delay factors for carriage returns.) This parameter is indicated by one of the numbers 0 through 8, with the meanings shown below:

0 = ASR 33 EIA-compatible HP 2749B (10-characters per second (cps)).

0 = ASR 35 EIA-compatible (10 cps).

1 = ASR 37 EIA-compatible (15 cps).

3 = Execuport 300 Data Communication Transceiver Terminal (10, 15, 30 cps).

4 = HP 2600A or DATAPOINT 3300 (10-240 cps).

5 = Memorex 1240 (10, 30, 60 cps).

6 = GE Terminet 300 or Data Communication Terminal Model B (DCT 500) (10, 15, 30 cps).

7 = Selectric (IBM 2741) CALL 360 Correspondence Code (14.8 cps).

8 = Selectric (IBM 2741) (PTTC/EBCD Code) (14.8 cps).

NOTE: Users at IBM Selectric terminals are directed to the NOTE under the discussion "Initiating Sessions," later in this section.

If the *termtype* parameter is omitted, and a terminal is used, a default of 0 is assigned. (Optional parameter for ASR 33 or ASR 35 terminals; Required parameter for all others, to ensure correct listings.)

cpulimit

Maximum central-processor time permitted for the job, entered in seconds. If this limit is reached during the job, the job is aborted. If a question mark is entered, no limit applies. The maximum limit entry is 32767. If omitted, an installation-defined limit applies. (Optional Parameter.)

NOTE: The cpulimit *parameter should not be confused with the central-processor time limit defined for groups and accounts. The* cpulimit *parameter (or its default value) applies to each individual job, and aborts the job as soon as it is exceeded. The central-processor time limit for groups and accounts, however, limits the total central-processor time used by all jobs that have run under the particular group and account--it is checked only at log-on time, and can only terminate access at that time; it will never cause a job in process to abort.*

executionpriority

The priority (subqueue) desired for MPE/3000 command interpretation, and also the default priority for all programs within the job. For users with *standard capabilities*, this may be CS, DS, or ES, indicating the priorities (subqueues) described in Sections II and VIII. For users with optional capabilities, this may be any of the priorities described in Sections II and VIII (depending on the maximum subqueue priority permitted this user). If the priority specified exceeds that permitted for this user, the highest priority possible below BS applies. If this parameter is omitted, CS is assigned by default. (Optional Parameter.)

selectionpriority

The relative priority to be used for selecting this job when several jobs of equal *executionpriority* are ready to begin processing. This parameter is one or two digits, ranging from 0 (lowest priority) to 15 (highest priority). The default value is 8. Note that this parameter merely supplements other criteria used to determine final selection. Additionally, other factors being equal, candidates of equal priority are serviced on a first-in, first-out basis. (Optional Parameter.)

outputclass

A particular device (such as a specific line printer) to be used for listing output. This parameter is a device class name or a logical device number (as defined in Section V), and may only be entered by users who have the *non-sharable input/output device* file-access attribute. If this parameter is omitted, the device assigned is the one defined (during system configuration) as the default job/session listing device that always corresponds to the user's current input device. Magnetic tape units cannot be accepted as the output device for a job. (Optional Parameter.)

The *upass*, *apass*, and *gpass* parameters, when included, must be separated from their preceding parameters with a slash. (The slash should not be preceded nor followed by blanks.) When the job list device is not the job input device, passwords are not printed on the job listing.

EXAMPLE:

This job is named ALPH22, and is entered under the user name RJOHNSON, account name ACCT1003, and the group GPA2. The group password MG03 is used. A central processor time limit of 300 seconds is entered. An execution priority of DS is requested. For all other parameters, default values apply. Notice the continuation character (&) at the end of the first line.

```
:JOB ALPH22,RJOHNSON.ACCT1ØØ3, GPA2/MGØ3; TIME=3ØØ; &
:PRI=DS
```

If the central-processor time already accumulated by previous jobs exceeds the total time allotted for the user's account or group, the job is rejected upon submission. (Note that these account/group limits are checked only at the start of a job.) Otherwise, the job runs until completed or until aborted because of an error or because the limit designated by the *cpulimit* parameter in the :JOB command is reached.

All MPE/3000 commands encountered during job processing are listed on the job listing device (typically, a *printer*). Acceptance of the job is indicated by information output in the following format, immediately after the listing of the :JOB command.

JOB NUMBER = #Jnnnnn
date, time
HP32000v.uu.ff

nnnnn Job number assigned by MPE/3000 to uniquely identify the job to the system. This may range from one to five digits long. The user can reference the job in subsequent commands by this number or by the job identification specified through the :JOB command:

[jobname,] username.acctname

date Current date (day-of-week, month and day, year)

time Current time (hours:minutes am/pm)

v MPE/3000 version level.

uu MPE/3000 update level.

ff MPE/3000 fix level.

EXAMPLE:

If the job in the previous example had been assigned the job number 7, and had been submitted at 3:23 p.m. on January 31, 1972, the job output listing would appear as follows (with the beginning of the acceptance message shown here by an arrow):

```
:JOB ALPH22,RJOHNSON.ACCT1ØØ3, GPA2/MGØ3; TIME=3ØØ; &
:PRI=DS
JOB NUMBER = #J7
MON,JAN 31, 1972, 3:23 PM
HP32ØØØB.Ø2.Ø8
```

Terminating Batch Jobs

The :EOJ command terminates the batch job. This command has no parameters and is written simply as

```
:EOJ
```

When this command is encountered, MPE/3000 acknowledges job termination by printing the following information on the job listing

```
CPU(SEC) = cputime
ELAPSED (MIN) = elapsedtime
date, time
END OF JOB
```

cputime Total central processor time used by job, in seconds, charged to account and group.

elapsedtime Total wall-clock time between the beginning and end-of-job, in minutes. This value is *not* charged to the account and group.

date Current date (day-of-week, month and day, year)

time Current time (hours:minutes am/pm)

If no :EOJ command is included to terminate the current job, the next :JOB command will have one of the following effects:

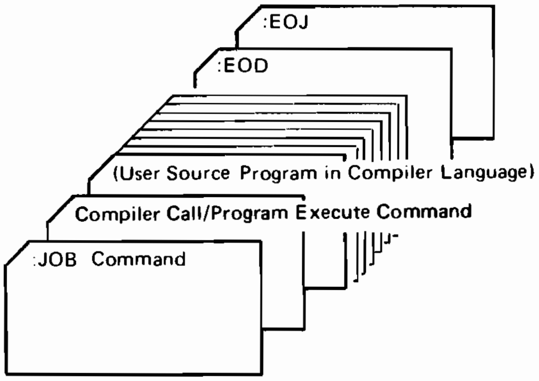
1. If read by the MPE/3000 Command Interpreter, it will terminate the current job and start a new one.
2. If read by a user's program, it will signal an end-of-file to the program but will be ignored by MPE/3000. (Because this can occur, the user should be sure that he terminates his job with an :EOJ command.)

Typical Job Structures

All batch jobs must begin with a :JOB command and terminate with an :EOJ command. Additionally, the end of each program input within a job should be indicated with an :EOD command, (written simply as :EOD). Beyond this, job structures can vary considerably from application to application.

EXAMPLE:

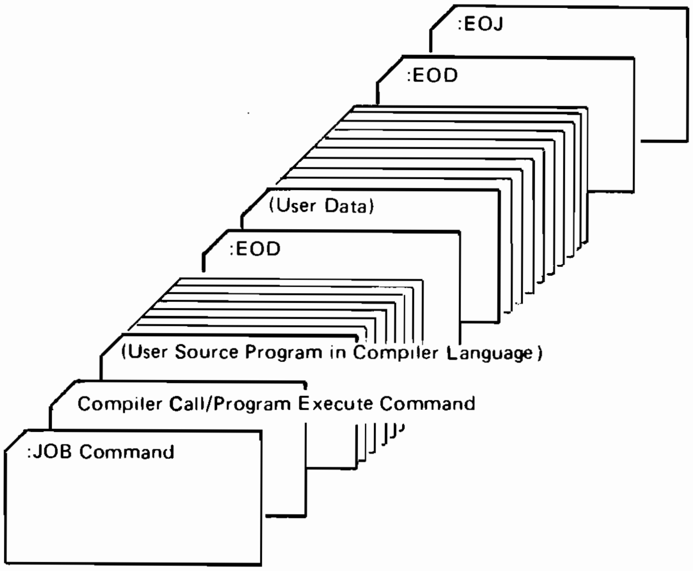
A job submitted through the card reader to compile and execute a program might appear as follows. (Note that an :EOD card denoting the end of the program must precede the :EOJ card.)


:EOJ
:EOD
(User Source Program in Compiler Language)
Compiler Call/Program Execute Command
:JOB Command

When data is included within a job, its end must also be indicated with an :EOD command. (Users writing language processors and other subsystems should be aware that MPE/3000, when reading input from the standard input stream, interprets *any* record beginning with a colon as the end-of-data, whether or not this record contains an :EOD command. The :EOD command is used as a data delimiter in general applications because it executes no *additional* functions.)

EXAMPLE:

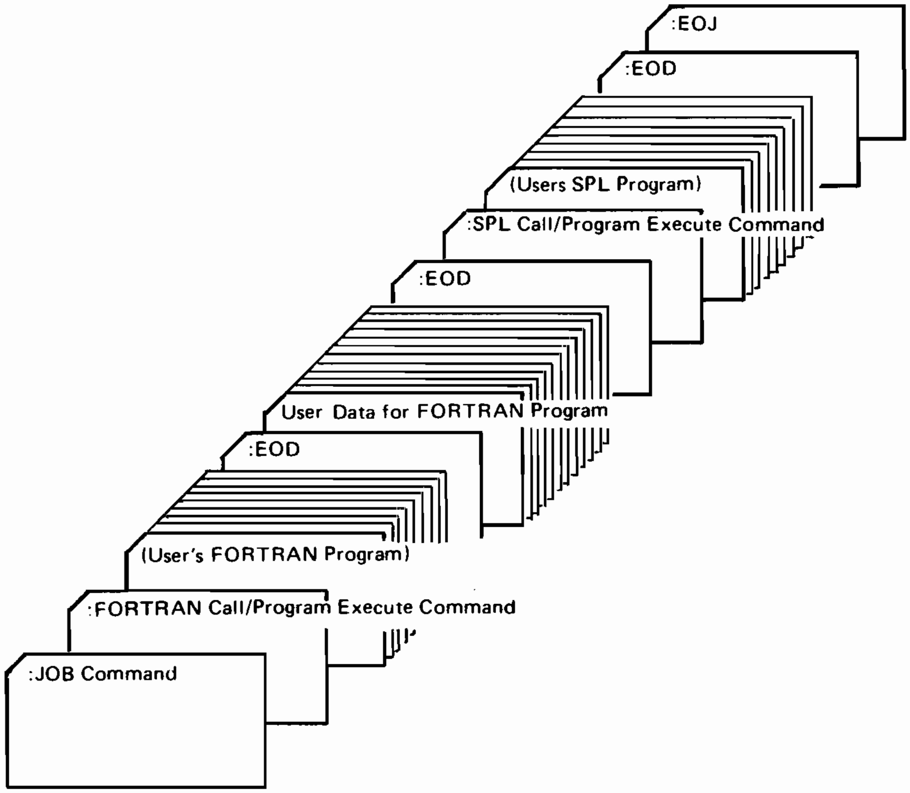
The job shown below includes data to be processed during program execution:


:EOJ
:EOD
(User Data)
:EOD
(User Source Program in Compiler Language)
Compiler Call/Program Execute Command
:JOB Command

Of course, jobs with much more elaborate structures are possible. Such jobs may contain several user programs, input to different compilers, obtaining input data from various sources, and transmitting output data to several devices:

EXAMPLE:

The following job contains: a FORTRAN/3000 program to be compiled and executed; data for that program; and an SPL/3000 program to be compiled and executed, with data input from a disc file.


:EOJ
:EOD
(Users SPL Program)
:SPL Call/Program Execute Command
:EOD
User Data for FORTRAN Program
:EOD
(User's FORTRAN Program)
:FORTRAN Call/Program Execute Command
:JOB Command

INTERACTIVE SESSIONS

The user initiates and terminates interactive sessions from a terminal. (Non-interactive devices cannot be used for this purpose.)

Initiating Sessions

The user initiates a session by turning the terminal on, dialing MPE/3000 (if the terminal is connected over switched telephone lines), pressing the *return* key to generate a carriage return and produce the first prompt character (colon), and entering the :HELLO command described below. This logging-on establishes contact with MPE/3000 and allows the user to enter commands to compile and run programs or request other MPE/3000 standard capabilities. From this point on, the user is automatically prompted for each command (and thus does not type the colon). The session exists until the user issues a command to terminate it, or until one of the conditions described under "Premature Job or Session Termination" occurs.

NOTE: *Users at IBM 2741 Selectric Terminals (PTTC/EBCD Code or Call 360 Correspondence Code) must always type H or h as the first character of their input following the initial prompt character; this enables MPE/3000 to determine (from this character's bit pattern) the type of Selectric being used.*

Users at IBM 2741 Selectric Terminals (CALL/360 or PTTC/EBCD code) should note that any CALL/360 or PTTC/EBCD character that does not have an equivalent ASCII character is ignored on input.

Users at IBM 2741 Selectric Terminals (Call 360 Correspondence Code) will receive a not-equals sign (≠) as the first prompt character.

The user enters the :HELLO command in the following format

```
:HELLO [sessionname,] username[/upass].acctname[/apass] [,groupname[,gpass] ]
       [;TERM = termtype]
       [;TIME = cpulimit]
       [;PRI = executionpriority]
       [;INPRI = selectionpriority]
```

The parameters have the meanings noted below.

NOTE: *The sessionname, username, upass, acctname, apass, groupname and gpass parameters are all names that can contain up to eight alphanumeric characters, beginning with a letter.*

sessionname — An arbitrary name used in conjunction with the *username* and *acctname* parameters to reference this session in other commands. If omitted, a null *sessionname* is assigned. (Optional parameter.)

username
upass
acctname
apass
groupname
gpass
termtype
executionpriority
selectionpriority — The same definitions noted under the corresponding parameters in the :JOB command, applying to *sessions* rather than jobs.

cpulimit — The same definition noted for *cpulimit* in the :JOB command, applying to *sessions*, and with this exception: if the *cpulimit* parameter is omitted, *no* limit applies.

The *upass*, *apass*, and *gpass* parameters, when included, must always be separated from their preceding parameters by a slash. (This slash should not be bounded by blanks.)

EXAMPLE:

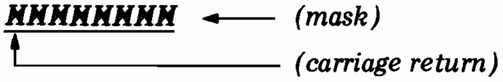
In the following command, the user establishes a session named INTER3, under the user name JONES, account name ACT20, and GROUP 3. The account password of PASS is also entered.

```
:HELLO INTER3,JONES.ACT20/PASS, GROUP3
```

When a user is beginning a session through a terminal connected over switched telephone lines (a dial-up terminal), he must specifically log on within a period defined during system configuration. Otherwise, his terminal is disconnected and he must dial MPE/3000 again.

When entering a session through certain full duplex terminals, the user can suppress the printing (echoing) of passwords at his terminal by temporarily requesting half-duplex mode (by pressing the *ESC* and ; keys). (The user returns to full-duplex mode by pressing the ESC and : keys.) Where the terminal does not permit dynamic half-duplex operation, the user can conceal the passwords simply by omitting them from the command; in this case, after the :HELLO command is echoed, MPE/3000 outputs a corresponding request for each required password, directs the carriage to skip one line, prints an eight-character mask, and returns the carriage to the beginning of the line, as shown:

$$\left\{\begin{array}{l}\underline{\textit{USER}}\\ \underline{\textit{ACCOUNT}}\\ \underline{\textit{GROUP}}\end{array}\right\}\quad \underline{\textit{PASSWORD?}}$$



The user then enters the password. The password is echoed on the input/listing device, but is printed on top of the mask to ensure privacy.

EXAMPLE:

Suppose that a group password of ORG was required, but not entered, in the command shown in the previous example. The user would be prompted for the password, and would enter it (over the mask) as follows:

```
:HELLO INTER3,JONES.ACT20/PASS, GROUP3
GROUP PASSWORD?
MMMMMMMM
```

For sessions in which input and listing output are generated on separate devices, MPE/3000 omits the password entered.

Upon initiation of the session, the following information is transmitted to the terminal:

SESSION NUMBER = #Snnnnn
date, time
HP32000v.uu.ff

nnnnn Session number assigned by MPE/3000 to uniquely identify the session. This may range from one to five digits long.

date Current date (day-of-week, month and day, year)

time Current time (hours:minutes am/pm)

v MPE/3000 version level.

uu MPE/3000 update level.

ff MPE/3000 fix level.

EXAMPLE:

Suppose that the session in the previous example is accepted by MPE/3000 under the session number 512 at 12:37 p.m. on May 5, 1972. The following information would be output. (Remember, the mask is printed before the password may be entered.)

```
:HELLO INTER3,JONES.ACT20/PASS, GROUP3
GROUP PASSWORD?
MMMMMMMM
SESSION NUMBER = #S512
FRI,MAY 5, 1972,  12:37 PM
HP32000B.02.08
```

If the connect time or the central processor time limit allotted to the account or group specified by the :HELLO command has been exceeded by previous sessions or jobs, the current session is rejected at log-on. If these limits expire during the session, however, the session is allowed to continue until terminated by the user; the next session under this account or group, however, will be rejected at log-on.

If the user enters an illegitimate :HELLO command, the following error message appears:

```
:ERR 8
```

At this point, a request to re-enter the :HELLO command appears. (The erroneous element in the previous :HELLO command is not identified.)

NOTE: *A system power or line failure automatically results in the permanent termination of sessions; the sessions must be reinitiated from the beginning.*

Interrupting Program Execution Within Sessions

The user can interrupt execution of a program or subsystem at any point in a session by striking the BREAK key. (However, the BREAK key does not interrupt execution of MPE/3000 commands.)

This returns control to the MPE/3000 Command Interpreter, allowing the user to issue commands to abort the program (without disrupting the session), perform various file or utility operations, or resume the program.

To abort the program, the user enters this command:

```
:ABORT
```

To request an operation (such as creating or deleting a file), the user enters the command for that operation. File and utility commands are executed immediately. Some commands, however, require aborting of the user's program before they can be executed. When the user enters such a command during a program break, MPE/3000 prints:

```
ABORT?
```

The user responds by entering *YES* (to abort the current program and execute the command) or *NO* (to return control to the Command Interpreter).

To resume a program at the point where it was interrupted, the user enters:

```
:RESUME
```

If a read operation was pending for the program on the terminal when the break occurred, the following message is output:

```
READ PENDING
```

The user must satisfy the pending read before program execution can resume. (All characters entered before the break are retained.)

NOTE: The :ABORT and :RESUME commands are legitimate only during a break.

Terminating Sessions

To terminate a session, the user enters the following command:

```
:BYE
```

MPE/3000 acknowledges termination by printing the following information:

```
CPU (SEC) = cputime
CONNECT (MIN) = connecttime
date, time
END OF SESSION
```

cputime = Total central-processor time used by the session, in seconds, logged against group and account.

connecttime Total wall clock time between log-on and log-off, in minutes, logged against group and account.

date Current date (day-of-week, month and day, year)

time Current time (hours:minutes am/pm)

EXAMPLE:

Typical terminal output resulting from logging off:

```
:BYE

CPU (SEC)=4
CONNECT (MIN)=2
THU,MAY 7, 1973,   12:56 PM
END OF SESSION
```

For accounting purposes, the connect time is rounded upward to the nearest minute, and the central processor time is shown in seconds. Both sums are added to the usage counters for the session's account and group.

If the user hangs up the receiver at a dial-up terminal, an implicit :BYE command is issued and the session terminates.

Typical Session Structure

All sessions must begin with a :HELLO command. They typically terminate with a :BYE command. The end of data within a session must be indicated with the :EOD command.

If a user issues a second :HELLO command within his current session, this results in an implicit :BYE and :HELLO sequence - that is, it terminates the current session and starts another one.

The structure of sessions varies with the application. One example follows.

EXAMPLE:

```
:HELLO (Session, user, and account identification.)
SESSION NUMBER = #Snnn
(date) (time)
HP3200v.uu.ff

:(MPE/3000 Command to call BASIC Interpreter.)
     .
     .
     .
(User Program in BASIC.)
(User Command to exit from BASIC Interpreter.)
:BYE
CPU (SEC) = cputime
CONNECT(MIN) = connecttime
date, time
END OF SESSION
```

READING DATA FROM OUTSIDE STANDARD INPUT STREAM

Users can designate, for a job or session, input consisting only of data to be read from a non-sharable, auto-recognizing device that is *not* the standard input device for that job or session. (An auto-recognizing device is one that automatically accepts the :JOB, :HELLO, or :DATA commands.) This designation is typically made to read a deck from a card reader while in session mode. Indeed, it is required for every data deck not imbedded in the standard input stream ($STDIN).

To designate the data for the job or session, the user must precede the data with the :DATA command and terminate it with the :EOD command. The :DATA command implicitly initiates communication with MPE/3000 and thus is the only command that is not entered within a formally-initiated job or session. (Note that the :DATA command does not apply to data from non-auto recognizing devices, or data imbedded in the job or session input stream.)

The :DATA command format is

:DATA [jobname,] username[/upass].acctname[/apass] [;filename]

(A complete job or session identification.)

jobname The name of the job or session that is to read the data. (Optional parameter.)

username The user's name, as established in MPE/3000 by the user with Account Manager Capability. (Required parameter.)

upass The user password.(Required if user has a password.)

acctname The name of the account, as established by the user with System Manager Capability. (Required parameter.)

apass The account password.(Required if account has a password.)

filename An additional qualifying name for the data that can be used by the job or session to access the data. It may be used, for example, to distinguish two separate data decks from different card readers read by the same program. If *filename* is omitted, no such distinguishing name is assigned. (Optional parameters.)

The data can *only* be read by a job or session that has the same identity:

[jobname,] username.acctname

The data exists in the system until read by the job/session.

If the *filename* parameter is omitted from the :DATA command, then the data can be read by *any* access from the job with the corresponding identity.

If the job attempts to read data but omits the file *formaldesignator* when opening the data file, any file preceded by a :DATA command referencing that job's identity will satisfy the request.

The *jobname*, *username*, *acctname*, and *filename* parameters are all names that can contain up to eight alphanumeric characters, beginning with a letter.

EXAMPLE:

To designate a card data file for a session identified as JOBSP, BLACK.ACTSP, the user submits the following :DATA command on a card preceding the data deck:

```
:DATA JOBSP,BLACK.ACTSP; FILESP
•
•
•
(USER DATA)
•
•
•
:EOD
```

The session can access the data in the card file by specifying the card reader (either by device class name or logical device number) and the filename FILESP.

PREMATURE JOB OR SESSION TERMINATION

A *job* is terminated before an :EOJ command is encountered, if any of the following events occur:

1. The Console Operator issues a command to terminate the job.
2. MPE/3000 aborts the job because an error occurred in the interpretation or execution of an MPE/3000 command.
3. MPE/3000 aborts the job because a subsystem error was encountered.
4. A program within the job is aborted.
5. A second :JOB command or a :DATA command is encountered. It will be executed, resulting in job termination (unless it is read as part of a program's data input from a file/device, in which case it signals an end-of-file tó the program but is ignored by MPE/3000).
6. The central processor time limit specified in the *cpulimit* parameter of the :JOB command (or its default value) was exceeded.

Events 2 through 4 (above) place the job in an *error state* (by setting the error state flag). Normally, this results in job termination. But, if the user anticipates an error as a result of a specific command, he can override premature termination by using the :CONTINUE command before that command. The :CONTINUE command format is

:CONTINUE

The :CONTINUE command permits the job to continue even though the following command results in an error (in which case the error state indication is re-set).

EXAMPLE:

Suppose the user submits a job to execute three programs (with the :RUN command), but anticipates a probable terminating error in the first program. To continue the job and execute the second and third programs, he uses the :CONTINUE command as follows:

```
:JOB JNAM,UNAM.ACCTNAM
:CONTINUE
:RUN PROG1
:RUN PROG2
:RUN PROG3
:EOJ
```

A session may be terminated before a :BYE command is encountered, if any of the following events occur:

1. The Console Operator issues a command to terminate the session.

2. A second :HELLO command, or a :JOB or :DATA command, is encountered in the session input stream. This will implicitly result in a :BYE command, terminating the current session.

3. The central processor time limit specified in the *cpulimit* parameter of the :HELLO command was exceeded.

SECTION IV
System Generation and Back-up

Hewlett-Packard furnishes MPE/3000 to the customer as a set of prepared program files on magnetic tape. In this initial configuration (generally performed at the factory by HP), the following are included:

- One user with System Manager and System Supervisor Capabilities (plus *all* other standard and optional capabilities from the MPE/3000 Capability Set). This user is identified by the user name MANAGER.
- One System Account (identified by the account name SYS).
- One Public Group (identified by the group name PUB, belonging to the System Account SYS).
- All files necessary to run MPE/3000. (These include all program files required, plus one USL for external interrupts and one segmented library named SL.PUB.SYS.)

Once the system is started (as described in Section V), the user with System Supervisor Capability can call the MPE/3000 Configurator program to

1. Re-configure MPE/3000 by interactively creating a modified version of the running system on magnetic tape. This permits the user to add new software; add or remove input/output devices; change system table sizes, system disc allocation, or segment size limits; and alter scheduling subqueue quanta. (Software that runs as a subsystem of MPE/3000, such as SPL/3000, FORTRAN/3000, COBOL/3000, or EDIT/3000 is provided on a separate reel of tape and is installed using the :RESTORE command, described in Appendix H.)

2. Copy the currently-running system to magnetic tape. This is done to provide back-up in case the running system is destroyed by a hardware or software failure, or to transfer the system to another installation.

3. Copy the currently-running system, the current account/group/user structure, and all files (or only those files most recently changed) to magnetic tape. This is done to provide back-up in case the information on the discs is destroyed by a hardware or software failure.

NOTE: A duplicate of the running system on tape is the only way that a user can provide back-up—if no such tape exists and a system failure occurs, the user must contact an HP field representative for help. (Also, see Appendix G.)

Regardless of whether the Configurator is used to re-configure MPE/3000 or to make an identical copy of it, the *running system* is not altered; any changes made apply only to the *copy* of the system written to tape. (Only magnetic tape, and not disc, can be used for this output.)

MODIFYING MPE/3000

When using the Configurator to modify MPE/3000, the user requires certain background information. The information he needs depends on the type of changes he wishes to make, as noted below. (Changes are made through an interactive dialogue between the user and the Configurator; the step number of the dialogue where the change is made is noted in the heading, below, that pertains to that change.)

Changing Main Memory Size [2]

MPE/3000 runs on HP 3000 Computers with main-memory of the following sizes:

32K (K = 1024 words)

48K

64K

The size specified by the user during configuration should be the actual size of main-memory delivered with the machine on which this MPE/3000 configuration is to run. This entry is required so that other configuration parameters (such as table sizes) that depend on main-memory size can be set up correctly.

Changing Input/Output Device Configuration [3]

Every physical input/output device in the system is identified by a unique logical device number, ranging from 1 to 255. Input/output configuration consists of specifying this number and various other characteristics for each such device. Some of these characteristics (such as DRT entry number and device unit number, described below) are determined by physical hardware connections made prior to system generation. Other characteristics (such as whether a device is interactive or duplicative, whether it can accept jobs and sessions, and the device class to which it belongs) are user options. When the user is deleting or re-specifying devices already on the system, he can determine the characteristics of these devices by requesting a Device Characteristics Listing during the re-configuration process. When he is adding a new device, he must know the hardware-dependent characteristics of the device and must also carefully determine those characteristics that are user options, as noted below. The characteristics that must be specified for each device are

1. Logical Device Number

   The logical device number is the value by which the MPE/3000 File Management System recognizes a particular device. For each device, this is a unique number ranging from 1 to 255, determined by the user.

2. Device Reference Table (DRT) Entry Number

   Every device on the system is cabled to a device controller. (A particular controller may serve more than one device of the same type.) For every controller, there is an entry in the DRT in main-memory that contains pointers to the driver and interrupt programs that serve the controller (and its devices). Because each DRT entry is four words long, the size of the DRT (in words) is

   4 x (Total number of controllers)

   The DRT is located in fixed-memory locations beginning at octal address 14. The maximum upper limit for the DRT is location 1777_8, thus limiting the maximum number of four-word DRT entries to 253 (decimal).

   Since each DRT entry is always four words long, it is convenient for the hardware to map controllers to DRT addresses simply by multiplying by four. Since the DRT begins at location 14, the lowest controller (DRT entry) number is 3 ($14_8/4_8 = 3_8$). (DRT entry numbers 0, 1, and 2 do not exist.)

   When re-configuring the system, the user needs to know the highest DRT entry number that can be assigned to a device. He determines this by adding three to the total number of controllers planned. This value may not exceed 255.

   The user also needs to know the DRT entry number of any device that he is adding or deleting. This is a hardware-dependent value, ranging from 3 to 255. It is determined by a set of jumpers on the device controller board.

3. Unit Numbers

   When a controller services only one device, that device is generally assigned a unit number of 0 (recognized by the associated driver). When the controller serves more than one device, each device is assigned a unique unit number (with respect to that controller) to distinguish it from others cabled to the same controller. The unit number of any device is a hardware-dependent characteristic determined when the device is physically connected to its controller. The value ranges from 0 to a maximum number determined by the type of device controller.

4. Device Type

This number determines the type of device, where

0	=	Moving-Head Disc
1	=	Fixed-Head Disc
8	=	Card Reader
9	=	Paper Tape Reader
16	=	Terminal
24	=	Magnetic Tape
32	=	Line Printer
33	=	Card Punch
34	=	Paper Tape Punch
35	=	Plotter

5. Device Sub-Type

This characteristic is specified for a device whose driver handles devices with different characteristics (such as hard-wired terminals versus terminals connected through a modem). It is a number ranging from 0 to 15, depending on the actual device referenced (as specified in Appendix D).

6. Corresponding Output Device

If the device can be used as a job/session input device, the user must specify (by logical device number *or* device class name) a device that will be recognized as the corresponding job/session list device. That is, all input read from the job/session input device is listed on that particular list device (or one of a set of devices, if a class name is specified).

7. Option to Accept Job and Session Input Stream

The user can optionally specify that this device can accept an input stream from a job or session — in other words, that it can accept the MPE/3000 :JOB or :HELLO commands, and thus serve as a job/session input device.

8. Option to Accept Data from Outside the Job/Session Input Stream

The user can optionally specify that a device can read data from outside the job/session input stream — in other words, that it can accept the :DATA command. (Typically, a device is designated to accept jobs/sessions, but not data external to a job/session.)

9. Interactive Option

The user can optionally specify that this device is a member of an *interactive pair.* An input device and a list device are said to be interactive if a real-time dialogue can be established between a program and a person using the list device as a channel for programmatic requests, with appropriate responses from a person using the input device. For example, an input file and a list file opened to the same terminal would be an interactive pair.

10. Duplicative Option

    The user can also specify that the device is a member of a duplicative pair; in such a pair, input from an input device is automatically duplicated on a corresponding list device. For example, a terminal upon which the information input is also output is duplicative.

11. Driver Name

    The name of the driver associated with the device/controller is specified. For standard devices supported by HP, appropriate driver names are found in Appendix D. For non-standard drivers supplied by the user, this is the name of the program file containing the driver; the name must contain from one to eight alphanumeric characters, beginning with a letter. (If the driver name is preceded by an asterisk (*), the driver will permanently reside in main-memory.)

12. Device Class Name

    The general class to which a device belongs must be specified. (This enables a user to request a device by class name, such as any disc or any tape unit.) These names are arbitrary, installation-dependent names that are left to the discretion of the System Supervisor. They consist of up to eight alphanumeric characters, beginning with a letter. A device can belong to more than one class, such as DISC and FHDISC. The only device class specifically referenced by MPE/3000 is DISC; it is the default device class for such operations as building files.

Changing System Tables and Queue Parameters [4]

The sizes of various system tables, queues, and other values can be changed to permit the user to make most efficient use of main-memory. The user can best determine what particular values are best for his installation by comparing the values supplied by HP in the initial configuration with those later suggested by his own operational experience. A table of default values (those normally provided initially by HP) appears in Appendix E.

Elements that can be changed are

1. The sizes of the following tables:

    Code Segment Table (CST)

    Data Segment Table (DST)

    Process Control Block (PCB) Table

2. The maximum number of entries in

    Input/Output Queue (IOQ)

    User Controller Process (UCOP) Request Queue

3. The number of

   Terminal Buffers in the system. (Each buffer is 36 characters long.)

   Input/Output Control Blocks (IOCB's) in the system. (Each buffered input/output requires an IOCB.)

4. The number of words in the interrupt control stack (ICS).

5. The maximum number of concurrent time-out requests allowed for the system clock.

Changing Miscellaneous Values Relating to Log-On Time, RIN's and Jobs Allowed in System and Execution [5]

The user can change

- The currently-assigned global resource identification numbers (RIN's)
- The number of RIN's available in the RIN pool
- The maximum number of global RIN's available
- The time a user is allowed to successfully complete logging-on to the system when initiating a session
- The maximum number of jobs allowed on the system
- The maximum number of jobs allowed in execution at one time
- The default central-processor time-limit for jobs

Changing Logging Characteristics [6]

The user can change

- The elements (types of entries) being logged in the system log file. This includes disabling or enabling the logging facility itself.
- The size of the records in the log file.
- The size of the log file.

Changing Disc Allocation [7]

The user can alter the maximum number of disc sectors available for virtual memory. Note that these sectors are used for data-segment swapping only, since code segments are read directly from files and need not be written back to disc.

The maximum number of disc sectors available for the system directory can also be changed. This directory contains the addresses of all files in the system, plus certain accounting information for accounts and groups.

The most efficient values for virtual memory and system directory sizes depend on the installation and its use. Generally-recommended values are shown in Appendix E.

Changing Scheduling Queue [8]

All processes competing for access to the central processor access it through the MPE/3000 Master Scheduling Queue. (This queue is described in general in Section II and in greater detail in Section VIII.)

The user can change the time-quantum, in milliseconds, allowed by any circular subqueue (CS or DS).

Changing Segment Limits [9]

The user can change the limits on code and data segments, as follows:

1. Maximum number of words allowed in any code segment.

2. Maximum number of code segments per process.

3. Maximum number of words allowed in any user process' stack.

4. Maximum number of words allowed in any extra data segment.

5. Maximum number of data segments per process.

6. Default number of words initially assigned for a user's stack (Z-Q) area (when the user specifies no value).

Generally-recommended values can be found in Appendix E.

Changing System Programs [10] and Segments in the System Library [11]

The user can replace system program files. He can also delete, add, and replace code segments in the System Library. During configuration, the user can request a list of all code segments currently in the System Library.

USING THE :SYSDUMP COMMAND

The user can either re-configure or copy his running MPE/3000 system onto magnetic tape by entering the :SYSDUMP command through his terminal during a session. This command can be entered at any time.

> *NOTE: To enter this command, the user requires the System Supervisor Capability.*

When the :SYSDUMP command is entered, the Configurator begins an interactive dialogue with the user by asking if he wants to make any configuration changes. If he responds NO, the Configurator copies the running system to tape. If he responds YES, the Configurator continues its dialogue with the user.

The Configurator's output consists of questions (ended by a question mark) and statements (ended by a period). The content of the questions generally indicates the type of answer required. To those questions requiring a simple positive or negative answer, the user responds with YES (or simply Y) or NO (or N, or simply a carriage return). Other questions contain values followed by a question mark; they normally quote an existing parameter value and ask whether the user wants to change it. To retain the quoted value, the user enters a carriage return. To change the value, he enters the new value desired. In any case, the user must always conclude an entry with a carriage return to transmit the entry to MPE/3000.

The :SYSDUMP command format is

:SYSDUMP dumpfile [,auxlistfile]

dumpfile A back-reference to a previous :FILE command that defines the file on which the modified or duplicate system is to be written. This back-reference *must indicate a magnetic tape file.* The formal file designator used by MPE/3000 is DUMPTAPE. The file open/close (FOPEN/FCLOSE) options provided by the :SYSDUMP command executor are:

NEW; REC=1024,,U,BINARY; SAVE; ACC=OUT

(All other file characteristics are supplied by MPE/3000 default.) (Back-referencing, formal file designators, file characteristics, and default options are all explained in the discussion of the :FILE command in *HP 3000 Multiprogramming Executive Operating System (03000-90005).*) (Required parameter.)

auxlistfile The actual file designator of the output file (device) to which all listings requested during the Configurator/User dialogue are written. The formal file designator used for this file by the :SYSDUMP command executor is SYSDLIST; the formal file designator used by the MPE/3000 Segmenter (when it is invoked to add or replace SL segments) is SEGLIST. The file open/close options provided are:

NEW; REC=,,V,ASCII; CCTL; ACC=OUT

If the *auxlistfile* parameter is omitted, the session list device (typically the user's terminal) is assigned by default. (Optional parameter.)

In addition to the names of the files to be used for the *dumpfile* and *auxlistfile* output, the user must specify what type of devices these files are to reside on. He does this by preceding the :SYSDUMP command with MPE/3000 :FILE commands defining each of the two files. Then, in the :SYSDUMP command, he precedes each filename with an asterisk to indicate a back-reference to the definitions in the :FILE commands. (The rules for writing and back-referencing :FILE commands are discussed in *Multiprogramming Executive Operating System (03000-90005).* An example is shown below, however, to illustrate the general way in which dialogue with the Configurator is initiated.)

EXAMPLE:

To begin dialogue with the MPE/3000 Configurator, a user enters the :SYSDUMP command shown below. At the end of the dialogue, the MPE/3000 configuration requested is copied to the file named DUMP. Any listings requested during the dialogue are output to the file named LIST. The first :FILE command specifies that DUMP is a magnetic tape file; the second :FILE command defines LIST as a line printer. (In these commands, the TAPE and LP parameters are device class names arbitrarily defined by a user during the previous configuration.)

```
:FILE DUMP; DEV=TAPE
:FILE LIST; DEV=LP
:SYSDUMP *DUMP,*LIST
```

CONFIGURATOR/USER DIALOGUE

The Configurator/User Dialogue proceeds as follows, with Configurator output shown verbatim in uppercase letters (underlined) and user input described in mixed upper-and lowercase letters.

Step No. **Dialogue**

0

```
ANY CHANGES?
```

To prepare for changes, enter YES.

To omit changes, and skip to Step 12, enter NO.

1

```
SYSTEM ID = HP32000B.<UU>.<FF>?
```

In this message, *uu* is the present update-level number and *ff* is the fix-level number.

To prepare for updating software for a new fix level, enter the new fix-level digits (ff). (These digits indicate the latest system fix provided by HP.)

Otherwise, enter a carriage return.

2

```
CORE SIZE=<XX>.?
```

In this message, *xx* denotes the present size of main memory.
To indicate the size of main-memory for the system for which MPE/3000 is being configured, enter one of the following values: 32, 48, and 64. This denotes memory size in a multiple of 1024 words.

3

```
I/O CONFIGURATION CHANGES?
```

To prepare for addition or deletion of input/output devices, enter YES.

To maintain the same input/output device configuration, and proceed to Step 4, enter NO.

3.1

```
LIST I/O DEVICES?
```

To print a list of input/output devices currently assigned to the system, enter YES.

To suppress this listing, enter NO.

Step No. **Dialogue**

If an input/output device listing is requested, it is displayed in tabular form, showing the following items for each device:

- Logical Device Number
- DRT Entry Number
- Unit Number
- Device Type, where

0	=	Moving-Head Disc
1	=	Fixed-Head Disc
8	=	Card Reader
9	=	Paper Tape Reader
16	=	Terminal
24	=	Magnetic Tape
32	=	Line Printer
33	=	Card Punch
34	=	Paper Tape Punch
35	=	Plotter

- Device Sub-Type.
- Record Width. (The default length of the physical records read or written by the device, in words.)
- Output Device. (The logical device number or device class name of the job/session list device corresponding to jobs/sessions entered on *this* input device. If this is not an *input* device, zero is output.)
- Mode, where

J = The device can accept jobs or sessions (:JOB or :HELLO commands).

A = The device can accept data external to job/session input stream (:DATA command).

I = The device is interactive.

D = The device is duplicative

Step No. Dialogue

- Driver Name. The name of the driver for this input/output device. (The driver resides permanently in main-memory if its name is preceded by an asterisk.)

- Device Classes. The classes to which this device belongs, defined by a System Supervisor user.

3.2 `HIGHEST DRT=<XX>.?`

In the output, *xx* is a value denoting the present highest DRT entry number that can be assigned to a device.

To change *xx*, enter the new value desired. If the highest-numbered device in the configuration is a device that uses more than one DRT entry (such as a terminal controller with one or two data set controllers), be sure to enter the *highest* of the DRT numbers.

To maintain the current *xx*, enter a carriage return.

3.3 `LOGICAL DEVICE #?`

To specify a device to be added or removed, enter the logical device number of that device.

To skip to Step 3.4, enter zero or a carriage return.

3.3.1 `DRT #?`

To add a device, enter its DRT entry number.

To remove a device and return to Step 3.3, enter zero.

3.3.2 `UNIT #?`

Enter the physical hardware unit number of the device, if the device shares its controller with other devices.

Otherwise, enter zero to continue.

Step No. **Dialogue**

3.3.3 TYPE?

Enter the device type (Appendix D).

3.3.4 SUB-TYPE?

Enter the device sub-type (Appendix D).

3.3.5 RECORD WIDTH?

Enter the record width for the device (Appendix D).

3.3.6 OUTPUT DEVICE?

If the device can be used as a job or session input device, enter the device class name or logical device number of the device to be used for the corresponding job/session listing device. (If a device class name is entered, any device of this class can be used as the listing device.)

If this is not a job/session input device, enter zero.

3.3.7 ACCEPT JOBS/SESSIONS?

To specify that this device can accept a job or session input stream, enter YES.

Otherwise, enter NO.

3.3.8 ACCEPT DATA?

To specify that this device can accept data external to a job or session input stream, enter YES.

Otherwise, enter NO.

Step No. **Dialogue**

3.3.9 `INTERACTIVE?`

To specify that this is an interactive device, enter YES.

Otherwise, enter NO.

3.3.10 `DUPLICATIVE?`

To specify that this is a duplicative device, enter YES.

Otherwise, enter NO.

3.3.11 `DRIVER NAME?`

Enter the name of the program file containing the driver for this device (Appendix D). For drivers written and supplied by the user, this name must contain from one to eight alphanumeric characters, beginning with a letter. (If the driver name is preceded by an asterisk, the driver will reside permanently in main-memory.)

3.3.12 `DEVICE CLASSES?`

Enter a list containing at least one device class name (up to eight alphanumeric characters, beginning with a letter). Class names are separated from each other by commas. These names are left to the discretion of the System Supervisor. They will be used in certain file commands when any member of a group of devices (such as any disc drive) can be referenced.

A device can belong to more than one class, such as DISC and FHDISC. When the device class list is complete, enter a carriage return to return to Step 3.3.

3.4 `LIST I/O DEVICES?`

To print a listing of the new input/output device configuration, enter YES. This list appears in the format described in Step 3.1.

To suppress the list, enter NO.

Set No. **Dialogue**

4 SYSTEM TABLE CHANGES?

To prepare for changing the CST, DST, or PCB tables, or other parameters relating to memory usage, enter YES.

To bypass these changes, and proceed to Step 5, enter NO.

4.1 CST=<XXX>.?

To change the size of the CST from *xxx* entries to another value, enter the new value.

To retain the current value, enter a carriage return.

4.2 DST=<XXX>.?

To change the size of the DST from *xxx* entries to another value, enter the new value.

To retain the current value, enter a carriage return.

4.3 PCB=<XXX>.?

To change the size of the PCB table from *xxx* entries to another value, enter the new value.

To retain the current value, enter a carriage return.

4.4 I/O QUEUE=<XXX>.?

To change the number of the input/output queue entries permitted from *xxx* entries to another value, enter the new value.

To retain the current value, enter a carriage return.

Set No. Dialogue

4.5 `TERMINAL BUFFERS=<XXX>?`

To change the number of terminal buffers in the system from *xxx*, enter the new value.

To retain the current value, enter a carriage return.

4.6 `IOCB=<XXX>?`

To change the number of input/output control blocks in the system, enter the new value.

To retain the current value, enter a carriage return.

4.7 `ICS=<XXX>.?`

To change the number of words in the interrupt control stack (ICS), enter the new value.

To retain the current value, enter a carriage return.

4.8 `UCOP REQUEST QUEUE=<XXX>.?`

To change the number of entries allowed in the User Controller Process Request Queue to another value, enter the new value.

To retain the current value, enter a carriage return.

4.9 `TIMER REQUEST LIST=<XXX>.?`

To change the maximum number of concurrent time-out requests for the system clock allowed, enter the new value.

To retain the current value, enter a carriage return.

Step No.	Dialogue
5	MISC CONFIGURATION CHANGES?

To prepare for the following miscellaneous configuration changes, enter YES:

- Listing and (optionally) deleting global resource identification numbers (RIN's) assigned to users.
- Number of RIN's available in the RIN pool.
- Maximum number of global RIN's available.
- Number of seconds allowed for logging-on.
- Maximum number of jobs allowed on the system.
- Maximum number of concurrent jobs allowed in execution.
- Default central-processor time-limit for jobs.

To bypass these changes and proceed to Step 6, enter NO.

5.1 LIST GLOBAL RINS?

To list the currently-assigned global resource identification numbers (RIN's), enter YES.

To suppress this listing, enter NO.

The listing consists of the RIN number and the name of the user and account to which it is assigned (for each RIN).

5.2 DELETE GOBAL RIN?

To prepare for deleting any of the currently-assigned global RIN's, enter YES.

To bypass deletion and skip to Step 5.3, enter NO.

5.2.1 ENTER RIN NUMBER?

To delete a currently-assigned global RIN, enter the RIN number.

This step is repeated until a carriage return is entered.

NOTE: *Since global RIN's are permanently assigned to users and the RIN numbers will be hard-coded into their programs, RIN's should be deleted with caution.*

For this same reason the most up-to-date RIN table (which resides on disc) is used when the system is cold-loaded, except in the case of a RELOAD. This implies that any changes to the RIN table occurring during a :SYSDUMP operation, including changes to the size of the table, only take effect when the tape produced by :SYSDUMP is cold-loaded using the RELOAD option (Section V).

Step No. **Dialogue**

5.2.2

```
LIST GLOBAL RINS?
```

To list the updated global RIN's (as in Step 5.1), enter YES.

To suppress the listing, enter NO.

5.3

```
# OF RINS  MIN=<YYY>,MAX=<XXX>.?
```

To change the number of RIN's available in the RIN pool, enter a new value for *xxx*. This value must be at least as great as *yyy*. (*yyy* is the maximum of 5 and the highest currently-assigned global RIN number.)

To maintain the current maximum, enter a carriage return.

5.4

```
# OF GLOBAL RINS USED=<YYY>,MAX=<XXX>.?
```

To change the maximum number of global RIN's available, enter a new value for *xxx*. Because of the current assignment of global RIN numbers, this must be at least as great as *yyy*.

To maintain the current value, enter a carriage return.

5.5

```
# OF SECONDS TO LOGON=<XXX>.?
```

To change the number of seconds allowed for logging-on, enter the new value.

To retain the current value, enter a carriage return.

Step No. **Dialogue**

5.6 `MAX # OF JOBS=<XXX>.?`

To change the maximum number of jobs allowed on the system, enter the new value.

To retain the current value, enter a carriage return.

5.7 `MAXIMUM # OF CONCURRENT RUNNING JOBS=<XXX>?`

To change the maximum number of jobs allowed in execution at one time, enter the new value.

To retain the current value, enter a carriage return.

5.8 `DEFAULT JOB CPU TIME LIMIT=<XXX>?`

To change the default value (in seconds) for limiting the amount of central processor time used by a single job, enter the new value. (A *zero* implies that jobs are not limited).

To retain the current value, enter a carriage return.

6 `LOGGING CHANGES?`

To prepare for changes to the logging characteristics of the system, enter YES.

To bypass such changes and proceed to Step 7, enter NO.

6.1 `LIST LOGGING STATUS?`

To print a list of the events that can be logged and whether or not they are currently being logged, enter YES.

To suppress the listing, enter NO.

6.2 `CHANGE STATUS?`

To prepare for changes to the logging status, enter YES. If no changes are desired, enter NO to skip to Step 6.3.

Step No. Dialogue

6.2.1

```
ENTER TYPE,ON/OFF?
```

The user should enter the type number of the event (defined below), a comma, and ON to signify that it is to be logged or OFF to signify that it is not.

The following events may be logged:

TYPE NO.	EVENT
1	Logging enabled
2	Job initiation
3	Job termination
4	Process termination
5	File close
6	System shutdown

NOTE: Event 1 must be ON for any logging to take place.

Step 6.2.1 is repeated until a carriage return is entered.

6.2.2

```
LIST LOGGING STATUS?
```

To list the updated logging status, respond with YES. To suppress the listing, enter NO.

6.3

```
LOG FILE RECORD SIZE (SECTORS)=<XX>?
```

To change the value of the log file physical record size, enter the number of sectors desired. This number determines the size of the buffer for entries in the log file. (A sector is equal to 128 words.)

To retain the current value, respond with a carriage return.

6.4

```
LOG FILE SIZE (RECORDS)=<XXX>.?
```

To change the maximum number of physical records permitted in the log file, enter a new value. The log file has 16 extents, so each extent will contain:

$$\left(\frac{\text{(log file size)}}{16} \times \text{(log file record size)}\right) \text{ sectors of disc space.}$$

To retain the present value, enter a carriage return.

Step No.	Dialogue

7 DISC ALLOCATION CHANGES?

To prepare for disc allocation changes, enter YES.

To bypass such changes and proceed to Step 8, enter NO.

7.1 VIRTUAL MEMORY=<XXXX>.?

To change the size of the area on disc used for virtual memory from *xxxx* sectors to another value, enter the new value.

To retain the current value, enter a carriage return.

7.2 DIRECTORY USED=<YYYY>,MIN=<ZZZZ>,MAX=<XXXX>.?

To change the maximum size of the directory from *xxxx* sectors, enter the new value; *yyyy* specifies the amount of directory currently used; *zzzz* specifies the minimum value to which *xxxx* can be set. (*zzzz* will often be greater than *yyyy* due to unused areas that are not at the end of the space allotted to the directory.)

To retain the present maximum size, enter a carriage return.

7.3 LIST VOLUME TABLE?

To list the disc volumes and their currently-assigned logical device numbers, enter YES. The listing is printed in the following format:

VOLUME	*LOG DEV #*
volname	*ldn*
.	.
.	.
.	.

In this listing, *volname* is a name of up to eight alphanumeric characters, beginning with a letter, identifying the volume; *ldn* is the logical device number assigned to that volume.

To suppress this listing, enter NO.

Step No. Dialogue

7.4 `DELETE VOLUME?`

To prepare to delete a volume, enter YES.

To bypass deletion and skip to Step 7.5, enter NO.

7.4.1 `ENTER VOLUME NAME?`

To delete a volume, enter the volume name. (When the name is entered, the question is repeated.)

Otherwise, enter a carriage return.

7.5 `ADD VOLUME?`

To prepare to add a volume, enter YES.

To bypass addition and skip to Step 7.6, enter NO.

7.5.1 `ENTER VOLUME NAME`

To add a volume, enter the volume name. (When the name is entered, the question is repeated.)

Otherwise, enter a carriage return.

7.6 `LIST VOLUME TABLE?`

To list the disc volumes and their currently assigned logical device numbers (as in Step 7.3), enter YES. In this listing, volumes just added (in Step 7.5) will have logical device numbers of zero.

To suppress this listing, enter NO.

Step No. **Dialogue**

8 `SCHEDULING CHANGES?`

To prepare for changes to the scheduling queue, enter YES.

To bypass these changes and proceed to Step 9, enter NO.

8.1 `TIME QUANTUM-CS SUBQUEUE=<XXX>.?`

To change the time-quantum of the CS subqueue (expressed in milliseconds), enter the new value.

To retain the current value, enter NO.

8.2 `TIME QUANTUM-DS SUBQUEUE=<XXX>.?`

To change the time-quantum of the DS subqueue, enter the new value.

To retain the current value, enter NO.

9 `SEGMENT LIMIT CHANGES?`

To prepare for changing the limits on code and data segments, enter YES.

To retain the current limits and skip to Step 10, enter NO.

9.1 `MAX CODE SEG SIZE=<XXXX>.?`

To change the maximum number of words allowed in any code segment from *xxxx*, enter the new value.

To retain the current value, enter a carriage return.

9.2 `MAX # OF CODE SEGMENTS/PROCESS=<XXX>.?`

To change the maximum number of code segments allowed any user process, enter the new value.

To retain the current value, enter NO.

Step No. Dialogue

9.3 `MAX STACK SIZE=<XXXXX>.?`

To change the maximum number of words allowed in any user stack from *xxxxx*, enter the new value. (A maximum value of 31232 is permitted.)

To retain the current value, enter a carriage return.

9.4 `MAX EXTRA DATA SEG SIZE=<XXXXX>.?`

To change the maximum number of words allowed in any extra data segment from *xxxxx*, enter the new value.

To retain the current value, enter a carriage return.

9.5 `MAX # OF EXTRA DATA SEGMENTS/PROCESS=<X>.?`

To change the maximum number of extra data segments that a process can have, enter the new value.

To retain the current value, enter a carriage return.

9.6 `STD STACK SIZE=<XXXX>.?`

To change the number of words initially assigned for a user stack (Z-Q area) by default (when the user specifies no value) at preparation time from *xxxx*, enter the new value.

To retain the current value, enter a carriage return.

10 `SYSTEM PROGRAM CHANGES?`

To prepare to replace a program belonging to the system, enter YES.

To proceed directly to Step 11, enter NO.

Step No. Dialogue

10.1 ENTER PROGRAM NAME, REPLACEMENT FILE NAME?

To replace a program belonging to the system, enter the name of the program, a delimiting comma, and the name of the program file which is to replace the program. (The replacement program need not be in the public group of the system account, PUB.SYS).

The question is repeated until a carriage return is entered.

11 SYSTEM SL CHANGES?

To prepare for changes to the system library (SL.PUB.SYS) enter YES. Otherwise, enter NO to skip to Step 12.

11.1 LIST LIBRARY?

To list the names of the code segments in the System Library and their entry-points and external procedures, enter YES.

To suppress this listing, enter NO.

11.2 DELETE SEGMENT?

To prepare for deleting a code segment from the System Library (SL.PUB.SYS), enter YES.

To proceed directly to Step 11.3, enter NO.

11.2.1 ENTER SEGMENT NAME?

To delete a code segment from the System Library, enter the name of that segment. (When the segment name is entered, the question is repeated.)

Otherwise, enter a carriage return.

11.3 REPLACE SEGMENT?

To prepare for replacing a code segment in the System Library, enter YES.

To proceed directly to Step 11.4, enter NO.

Step No. Dialogue

11.3.1 **ENTER SEGMENT NAME,USLFILE NAME [,S/C/P]?**

To replace a code segment in the System Library, enter the name of the segment; a delimiting comma; and the name of the USL file where the replacement segment can be found. Also, optionally, enter a delimiting comma followed by one of these three characters:

S To declare the segment to be a permanently-allocated system intrinsic segment (in virtual memory).

C To declare the segment to be a main-memory resident system intrinsic segment.

P To declare the segment to be a permanently-allocated user segment (in virtual memory). (This option requests the same function as the :ALLOCATE command, described in Section VIII.)

The question is then repeated.

Otherwise, enter a carriage return.

11.4 **ADD SEGMENT?**

To prepare for adding a code segment to the System Library, enter YES.

Otherwise, enter NO to skip to Step 11.5.

11.4.1 **ENTER SEGMENT NAME, USLFILE NAME [,S/C/P]?**

To add a code segment to the System Library, enter the name of the segment; a delimiting comma; the name of the USL file where the segment can be found. Also, optionally, enter a delimiting comma followed by one of these three characters:

S To declare the segment to be a permanently-allocated system intrinsic segment (in virtual memory).

C To declare the segment to be a main-memory resident system intrinsic segment.

P To declare the segment to be a permanently-allocated user segment (in virtual memory). (This option requests the same function as the :ALLOCATE command, described in Section VIII.)

The question is then repeated.

Otherwise, enter a carriage return.

11.5 LIST LIBRARY?

To list the updated system library, enter YES.

To suppress this listing, enter NO.

12 ENTER DUMP DATE?

To copy only the MPE/3000 system to tape, enter a carriage return; the dialogue skips to Step 13.

To copy the MPE/3000 system, the current accounting structure, and all files to tape, enter *0*. This tape can then be used to RELOAD the system (as described is Section V) if the information presently on disc is somehow destroyed. The tape can also be used with the :RESTORE command (described in Appendix H) to retrieve a lost file.

To copy the MPE/3000 system, the current accounting structure, and any files that were changed on or after a particular date, enter that date in the format *mm/dd/yy*. (In this format, *mm*, *dd*, and *yy* are sets of two decimal digits representing the month, day, and year, respectively.) This tape can be used in conjunction with other tapes to RELOAD the system following a disc failure, or to retrieve one or more files by using the :RESTORE command.

NOTE: Because files in use with write-access will not be copied, system back-up should be performed only when no users are logged onto the system.

12.1 LIST FILES DUMPED?

To obtain a list showing the name of each file copied, enter YES. To suppress this list, enter NO.

(A list showing the *number* (count) of files copied, the number of files not copied, the names of the files not copied, and the reasons why they were not copied is *always* provided.)

Step No. Dialogue

13 The system is now copied to tape (multi-reel files). It can then be loaded and initialized as directed in the next section of this manual.

Any file belonging to the system that was not copied for some reason will be noted. If that file was to replace a system program, the program name will follow in parentheses.

If a response other than a carriage return was entered in answer to the ENTER DUMP DATE? question in Step 12, the list and count of files will be provided as described in Step 12.1.

To denote termination of the Configurator/User Dialogue, the following message is printed:

```
END OF SUBSYSTEM
```

SECTION V
System Start-up, Modification and Shutdown

Any person with access to the MPE/3000 Console can start up MPE/3000, alter the current input/output device configuration, and shut down the system. (This person need not have System Manager or System Supervisor Capability; in fact, since he need not log-on to the system, he need not even have standard user capability.)

START-UP AND MODIFICATION

Start-up of MPE/3000 and re-configuration of the input/output devices on the system are done through a program called the MPE/3000 Initiator. The Initiator provides an option of four types of start:

- COOLSTART cold-loads the MPE/3000 System from the system disc. (This is the standard operating procedure when a system is routinely shut down at night and brought up again the next day.) All resident user files (including programs such as FORTRAN/3000, COBOL/3000, SPL/3000, and EDIT/3000 that run as MPE/3000 subsystems) are saved, but the operational environment present prior to the last shutdown is not retained. Thus, all jobs or sessions in progress at shutdown are lost.

- COLDSTART cold-loads the MPE/3000 System from magnetic tape, using the system files and input/output device configuration on that tape while retaining the user files, directory and accounting information, and assigned resource identification numbers (global RIN's) currently on disc. This allows modification of the system configuration while retaining the user's information. COLDSTART is commonly used to allow an installation to keep several cold-load tapes, each with a different configuration.

- UPDATE cold-loads the MPE/3000 System from magnetic tape, using the system files from that tape; the input/output devices, system configuration, directory, accounting information, and global RIN's from the system disc; and the user files from disc. This is the standard operating procedure used when starting the system with an updated MPE/3000 tape from HP or an MPE/3000 tape prepared for a different HP 3000 computer, and should be used *only* in those situations.

- RELOAD cold-loads the entire MPE/3000 System, including all system files and system and input/output configuration information, from magnetic tape. This option assumes that there is no information on disc. If any user files were dumped on the tape, the directory, accounting information, assigned global RIN's, and user files are restored to the disc from the tape. If no user files were dumped, a directory is created with the SYS account, PUB group, and MANAGER user. RELOAD is normally done when installing the system from the first MPE/3000 tape received from HP, or when restoring the system following a disc crash (from a tape generated by the user through a :SYSDUMP command (Section IV)).

  When reloading from multiple sets of tapes created by the :SYSDUMP command (Section IV), the first reel of the latest tape set should be used for cold-loading; it contains the up-to-date directory and accounting information. If not all of the files on the system are contained in this tape set, an additional set will be requested. Below are two examples of system back-up methods. (Also see Appendix G.)

  Method 1:

  This method minimizes the number of tapes and the amount of time needed to reload, but requires more time to dump the files. On Monday, the system is dumped using a dump date of 0. This dumps all of the files. On Tuesday, Wednesday, and on through Sunday, the system is dumped using Monday's date (the date of the previous complete dump) as the dump date. If the system had to be reloaded on, say, Friday, the user/operator would use Thursday's tape set first and then Monday's tape set. Thus, only two tape sets are required to reload all of the files.

  Method 2:

  This method minimizes the time taken to dump the system but requires more tapes and more time to reload. On Monday, all files are dumped as in Method 1. On Tuesday, the dump is performed using Monday's date as the dump date. On Wednesday, Tuesday's date is used, on through Sunday, when Saturday's date is used. If the system has to be reloaded on Friday, Thursday's tape set is used first, followed by Wednesday's, then Tuesday's, and then Monday's. This will take longer than Method 1, but the dump time is minimized since only those files changed in one day are dumped.

  Regardless of the method used, a user/operator would probably want to alternate between at least two groups of tapes, switching them on the day the complete dump is done (for instance, Monday).

No start-up option permits resumption of user batch jobs or interactive sessions interrupted by a system shut-down or hardware or software failure; such jobs must be re-initiated from the beginning by the user.

If a RELOAD is aborted for some reason, the next cold load also must be a RELOAD. If a COLDSTART or UPDATE is aborted, the next cold load must be a COLDSTART, UPDATE, or RELOAD. Any violations of this rule result in an error message and a halt.

All four start-up options allow the user to alter the input/output device configuration currently in effect. This is done through an Initiator/User Dialogue identical to Steps 2 through 3.4 of the Configurator/User Dialogue described in Section IV. (For information about the kind of information required in this dialogue, see Section IV.)

Disc Volume Organization

When the system is started, the initialization of blank disc packs, the writing of labels for them, and the renaming of volumes is accomplished by the MPE/3000 Initiator. (Formatting of blank disc packs, however, is done by a disc diagnostic program rather than by the MPE/3000 Initiator.)

Each disc connected to the system is known by a unique *volume name*. This name is kept in Sector 0 of the disc. A table of volume names, known as the *Volume Table*, is kept on the system disc; it maps each volume name into the logical device number of the disc on which the volume is mounted. This allows disc packs of similar type to be mounted on any disc drive (for instance, switching packs among drives between cold-loads), except that the system disc must always be in DRT Entry Number 5, Unit Number 0.

The *file directory* tells the volume on which a particular file is to be found, and its sector address on that volume. When the system is cold-loaded in any mode except RELOAD, the volume table that resides on the disc is used. The Initiator checks to ensure that all volumes defined previously are indeed mounted. This ensures that all files contained in the file directory are still there. A volume may be added but not deleted, since files defined in the directory may reside on that volume.

When the system is cold-loaded using the RELOAD option, the volume table on the cold-load tape is used. Volumes may be deleted or added, since each of the user files defined in the directory on the tape will be reloaded onto one of the available volumes.

Disc Error Recovery

When the system is first cold-loaded, it is assumed that the disc diagnostic has been used to format all discs and flag any defective tracks. When the MPE/3000 Initiator encounters a disc that does not have a valid MPE/3000 disc label, the operator is asked to give the disc a volume name. If the disc is the moving-head type, the operator is also requested to define the logical pack size of the disc (in cylinders); this is the amount of space available for use by the system and user files. The remaining portion of the disc may be used to re-assign alternate tracks when a disc error occurs. For moving-head discs, a track-by-track read is then initiated. Any tracks flagged as defective may be re-assigned on alternate tracks, or deleted (their space will not be available for use).

When a track-specific disc error occurs during running of the system, an entry is made in the Defective Tracks Table that resides on Sector 1 of the disc. The next time that the system is cold-loaded, the operator is informed of the error and requested to take action. If the disc is the moving-head type and alternate tracks are available, the defective track may be re-assigned. In any case, the track may be deleted (space made unavailable), recovered (entry removed from Defective Tracks Table) or ignored (entry left in Defective Tracks Table and the operator is prompted again on next cold-load). If this is not a RELOAD and any tracks were re-assigned or deleted, all files that were wholly or partially contained on those tracks are purged from the directory and a message is printed to this effect.

Disc errors in certain areas of the disc require special considerations:

- Tracks in the directory area may not be deleted or re-assigned except during a RELOAD, in which case the directory is moved.
- Tracks in the virtual memory area may not be re-assigned except during a RELOAD (which causes the virtual memory to be moved).
- Tracks in the alternate track area may not be re-assigned.
- Tracks in the reserved areas (location of label, Defective Tracks Table, bootstrap program, and disc free-space map) may not be deleted or re-assigned.
- Tracks in the system area (system tables, Initiator code, configuration tables, and so forth) may not be deleted or re-assigned during a COOLSTART.

Initiator/User Dialogue

The Initiator's output consists of questions (ended by a question mark) and statements (ended by a period). The content of the questions generally indicates the type of answer required. To those questions requiring a simple positive or negative answer, the user responds with YES (or simply Y) or NO (or N, or simply a carriage return). Other questions contain values followed by a question mark; they normally quote an existing parameter value and ask whether the user wants to change it. To retain the quoted value, the user enters a carriage return. To change the value, he enters the new value desired. In any case, the user must always conclude an entry with a carriage return to transmit the entry to MPE/3000.

To begin the Initiator/User Dialogue, the user follows these steps:

1. Mount the magnetic tape or disc containing the system and any associated new files and configuration data on the appropriate device. If the system is loaded from the system disc, that disc should be mounted on the disc drive specified by Device Reference Table (DRT) Entry Number 5, Unit 0. (Other discs can be mounted on whatever drives desired.) If the system is loaded from magnetic tape, that tape should be mounted on the tape unit specified by DRT Entry Number 6, Unit 0. The operator's console must be the terminal specified by DRT Entry Number 3.

2. Set the SWITCH REGISTER on the Computer Control Panel with the control-byte setting and DRT entry number of the device on which the system disc or system tape is mounted. (If the system is loaded from disc, set the switch register to octal 5. If the system is loaded from magnetic tape, set the switch register to octal 3006. (The control byte setting is 0 for disc and 6 for tape.)

The SWITCH REGISTER (Figure 5-1) consists of 16-bits; each bit is set *on* by placing the corresponding switch in the up position, or *off* by placing that switch in the down position. The control byte is represented by the leftmost byte (Bits 0-7); the DRT entry number, by the rightmost byte (Bits 8-15).

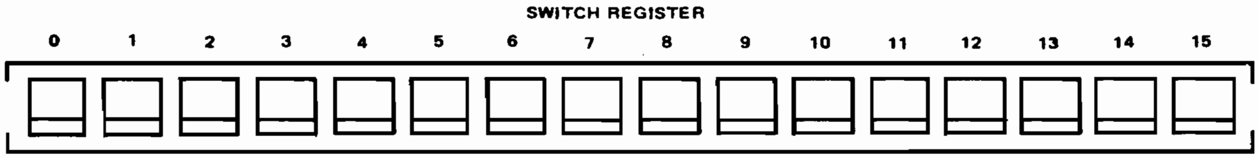


Figure 5-1. Switch Register

3. Press the following switches (on the Computer Control Panel) downward in the order shown:

   a. I/O RESET

   b. CPU RESET

   c. COLD LOAD

   This is the cold-load operation which reads into memory the MPE/3000 Initiator.

4. Press the RUN-HALT switch. The MPE/3000 Initiator enters execution, printing the following message on console:

```
HP 32ØØØB.<UU>.<FF>
```

In this message, *uu* is the present update-level number and *ff* is the fix-level number.

The Initiator next begins its dialogue with the user by printing the following information on the console. (In the dialogue shown below, output from the Initiator is shown verbatim in underlined upper-case letters, and input from the user is described in mixed upper and lower-case letters.)

If cold-loading is done from magnetic tape, the MPE/3000 Initiator begins its dialogue at Step 1. If cold-loading is done from disc, the COOLSTART option is assumed, and the dialogue begins with Step 4.

Step No. Dialogue

1

```
WHICH OPTION <COLDSTART/RELOAD/UPDATE>?
```

Enter the option desired. If RELOAD is chosen and user files exist on the tape, the Initiator proceeds to Step 2 (below). However, if RELOAD is chosen and *no* user files reside on the tape, the Initiator skips to Step 3. If either COLDSTART or UPDATE is selected, the Initiator skips to Step 4.

Step No. **Dialogue**

2 `WHICH OPTION <SPREAD/COMPACT/RESTORE/NULL>?`

Enter the desired RELOAD option, as described below. (A carriage return implies SPREAD.) The Initiator next skips to Step 4.

These options select the algorithm used to determine on which disc a file is placed during RELOAD, as follows:

SPREAD MPE/3000 attempts to place the file back on a disc of the same type and sub-type as the disc on which it was previously located (for instance, on a cartridge disc). If this fails, MPE/3000 attempts to place it on a disc of the same type (for example, a moving-head disc). If this fails, MPE/3000 attempts to place the file on any disc in device class *DISC*. If this fails, a message is printed and the file is not reloaded. In each of these attempts, the files are spread among similar discs, if possible. (Suppose, for example, that when the system was dumped, there was one cartridge disc that was full, and when it is reloaded there are two cartridge discs; in this case, each disc will be approximately half full.) The advantages of the SPREAD option are reduced disc seeking since files are spread out, and reduced fragmentation since the disc is re-packed. The disadvantage is that if the discs are nearly full, it may not be possible to restore all files that were previously stored on the discs. This situation is encountered very rarely; when it is, one of the other options may be used.

COMPACT MPE/3000 attempts to place the file back on the same volume from which it came. If this fails, the SPREAD option is used. The major advantage of COMPACT is that if there are no new deleted tracks and the same discs are used, reloading of the system is guaranteed, no matter how full the discs are. In addition, each disc is compacted within the area between deleted tracks (if there are n deleted tracks, there will be at most (n + 1) fragments). The disadvantage is that the discs may become disparately full.

RESTORE MPE/3000 attempts to place the files back on the same volume at the same locations from which they came. If this fails, MPE/3000 attempts to place the files anywhere on the volume from which they came. If this fails, the SPREAD option is used. The advantages to RESTORE are that it offers the same guarantee made in COMPACT for reloading the system, and that the same files that were previously using alternate tracks are still using them. The disadvantage is that no compacting of the discs is done (so that the same fragmentation still exists).

Step No. **Dialogue**

NULL MPE/3000 creates a null directory (as described in Step 3) and *no* user files are copied to the disc.

Note: In general, the SPREAD option should be used for reloading the system. If all files cannot be reloaded using this method, COMPACT should be used. RESTORE is useful only when fragmentation is not important but maintaining alternate track assignments is.

3

```
NO USER FILES FOUND - DO YOU WANT TO RELOAD?
```

This implies that when the system tape was created, no user files were dumped. To proceed with reloading the system, creating a null directory with only the SYS account, PUB group, and MANAGER user, enter YES. To return to Step 1, enter No.

4

```
LOAD MAP?
```

To request a map showing the correspondences between MPE/3000 segments, programs, and code segment table (CST) entries, enter YES. (The format of the map is shown in Figure 5-2, following this dialogue.)

To suppress this map, enter NO.

5

```
ANY CHANGES?
```

To proceed with changes to the current configuration, enter YES.

To maintain the current configuration, enter NO. If this is a RELOAD, the Initiator skips to Step 41. Otherwise, it skips to Step 25.

6

```
CORE SIZE=<XX>?
```

The value *xx* is the current size of main-memory for the system (in multiples of 1024 words). To change this value, enter one of the following values (specifying memory size in multiples of 1024 words): 32, 48, or 64.

Step No. Dialogue

7 <u>I/O CONFIGURATION CHANGES?</u>

To prepare for addition or deletion of input/output devices, enter YES.

To maintain the same input/output device configuration, and proceed to Step 25, enter NO.

8 <u>LIST I/O DEVICES?</u>

To print a list of input/output devices currently assigned to the system, enter YES.

To suppress this listing, enter NO.

If an input/output device listing is requested, it is displayed in tabular form, showing the following items for each device:

- Logical Device Number
- DRT Entry Number
- Unit Number
- Device Type, where

  0 = Moving-Head Disc
  1 = Fixed-Head Disc
  8 = Card Reader
  9 = Paper Tape Reader
  16 = Terminal
  24 = Magnetic Tape
  32 = Line Printer
  33 = Card Punch
  34 = Paper Tape Punch
  35 = Plotter

- Device Sub-Type
- Record Length. (The default length of the physical records read or written by the device, in words.)
- Output Device. (The logical device number or device class name of the job/session list device corresponding to jobs/sessions entered on *this* input device. (If this is not an *input* device, zero is output.)

Step No. Dialogue

- Mode, where

  J = The device can accept jobs or sessions (:JOB or :HELLO commands).

  A = The device can accept data external to job/session input stream (:DATA command).

  I = The device is interactive.

  D = The device is duplicative.

- Driver Name. The name of the driver for this input/output device. (The driver resides permanently in main-memory if its name is preceded by an asterisk.)

- Device Classes. The classes to which this device belongs, defined by a System Supervisor user.

9

```
HIGHEST DRT=<XX>.?
```

In the output, *xx* is a number denoting the present highest DRT entry number that can be assigned to a device.

To change *xx*, enter the new value desired. If the highest-numbered device in the configuration is a device that uses more than one DRT entry (such as a terminal controller with one or two data set controllers), be sure to enter the *highest* of the DRT numbers.

To maintain the current *xx*, enter a carriage return.

10

```
LOGICAL DEVICE #?
```

To specify a device to be added or removed, enter the logical device number of that device.

To skip to Step 23, enter zero or a carriage return.

11

```
DRT #?
```

To add a device, enter its DRT entry number.

To remove a device and return to Step 10, enter zero.

Step No. Dialogue

12 UNIT #?

Enter the physical hardware unit number of the device, if the device shares its controller with other devices.

Otherwise, enter zero to continue.

13 TYPE?

Enter the device type (Appendix D).

14 SUB-TYPE?

Enter the device sub-type (Appendix D).

15 RECORD WIDTH?

Enter the default physical record width for the device (Appendix D).

16 OUTPUT DEVICE?

If the device can be used as a job or session input device, enter the device class name or logical device number of the device to be used for the corresponding job/session listing device. (If a device class name is entered, any device of this class can be used as the listing device.)

If this is not a job/session input device, enter zero.

17 ACCEPT JOBS OR SESSIONS?

To specify that this device can accept a job or session input stream, enter YES.

Otherwise, enter NO.

18 ACCEPT DATA?

To specify that this device can accept data external to a job or session input stream, enter YES.

Otherwise, enter NO.

Step No. Dialogue

19 INTERACTIVE?

To specify that this is an interactive device, enter YES.

Otherwise, enter NO.

20 DUPLICATIVE?

To specify that this is a duplicative device, enter YES.

Otherwise, enter NO.

21 DRIVER NAME?

Enter the name of the program file containing the driver for this device (Appendix D). For drivers written and supplied by the user, this name must contain from one to eight alphanumeric characters, beginning with a letter. (If the driver name is preceded by an asterisk, the driver will permanently reside in main-memory.)

22 DEVICE CLASSES?

Enter a list containing at least one device class name (up to eight alphanumeric characters, beginning with a letter). Class names are separated from each other by commas. These names are left to the discretion of the System Supervisor. They will be used in certain file commands when any member of a group of devices (such as any disc drive) can be referenced.

A device can belong to more than one device class, such as DISC and FHDISC. When the device class list is complete, enter a carriage return to return to Step 10.

23 LIST I/O DEVICES?

To print a listing of the new input/output device configuration, enter YES. This list appears in the format described in Step 8.

To suppress the list, enter NO.

Step No. Dialogue

24 At this time, one or more of the following *messages* may be output, reflecting errors in the input/output device configuration. If any of these messages appear, the Initiator returns to Step 8.

```
DEVICE OF DIFFERENT TYPE RANGES IN CLASS <CLASS>
```

Device types (defined in Step 13) are divided into ranges for different kinds of devices:

Type Range	Class
0-7	Direct Access.
8-15	Serial Input.
16-23	Concurrent input/output.
24-31	Non-concurrent input/output.
32-39	Serial output.

Within a device *class*, all defined devices must have types which are in the same range; the above message indicates a violation of this rule.

```
HIGHEST DRT ALLOWED IS <DRT>
```

At least one device has been defined with a DRT number higher than the specified maximum (*drt*).

```
LOGICAL DEVICE <LDN> DOES NOT EXIST
```

A device (*ldn*) specified as an output device is not defined in the configuration.

```
MORE THAN ONE DEVICE FOR DRT <DRT> UNIT <UNIT>
```

More than one logical device has been defined with the same DRT, unit, and subtype.

```
NO DEVICE IN CLASS DISC
```

There must always be at least one device defined in the device class "DISC."

```
NO OUTPUT DEVICE FOR LOGICAL DEVICE <LDN>
```

Logical device *ldn*, which has been defined as a device which accepts jobs or sessions, does not have a corresponding job/session list device or device class assigned.

Step No. **Dialogue**

```
OUTPUT CLASS FOR DEVICE <LDN> NO LONGER EXISTS
```

The device class specified as the output (listing device) class for device *ldn* is not defined in the configuration.

Note: If all devices in a device class that is the output class for a specified device are deleted and then a new device in that class is added, this message is still output. It is then necessary to redefine the device which has this class as its output class.

```
SYSTEM CONSOLE MUST BE IN DRT 3
```

A terminal must be defined for DRT Entry Number 3.

```
SYSTEM DISC MUST BE IN DRT 5 UNIT Ø
```

A disc-type device must be defined for DRT Entry Number 5, Unit 0.

```
SYSTEM TAPE MUST BE IN DRT 6 UNIT Ø
```

A magnetic tape must be defined for DRT Entry Number 6, Unit 0.

25 At this time, the Initiator checks to ensure that all volumes defined in the previous cold-load are mounted. If all volumes were not found, the message *FOLLOWING VOLUMES NOT FOUND* is output, followed by a list of volumes that were defined but not mounted. The Initiator then proceeds to Step 26.

Note: One of the following messages may be printed, indicating that a disc device is not in the ready state:

```
LDEV #<LDN> NOT READY

DISC IN DRT #<DRT> UNIT Ø NOT READY
```

The operator should make the appropriate device ready. It is imperative *not to halt the system at this point, (or during any other portion of the Initiator/User Dialogue, unless the RELOAD option was selected), since the Initiator may be in the process of updating the volume labels. If the computer is halted, the message* FOLLOWING VOLUMES NOT FOUND *may be printed during the next cold-load, necessitating a RELOAD operation.*

If all volumes were found and the user had responded YES to the ANY CHANGES? question in Step 5, the Initiator skips to Step 28.

If all volumes were found and the user had *not* responded YES to the ANY CHANGES? question in Step 5, the Initiator skips to Step 41.

Step No. **Dialogue**

26 `LIST VOLUME TABLE?`

To list the disc volumes and their currently-assigned logical device numbers, enter YES. The listing is printed in the following format:

```
VOLUME          LOG DEV #

volname         ldn
   .             .
   .             .
   .             .
```

In this listing, *volname* is a name of up to eight alphanumeric characters, beginning with a letter, identifying the volume; *ldn* is the logical device number assigned to that volume. If *ldn* is 0, this indicates that the volume is not mounted.

To suppress this listing, enter NO.

27 `MOUNT CORRECT VOLUMES OR RELOAD`

Following this message, the system halts. The operator should either find the volumes defined on the previous cold load (listed in Step 25), mount them, and cold load the system again or cold load using the RELOAD option and redefine the disc volume configuration (as discussed below).

28 `DISC VOLUME CHANGES?`

To prepare for changes to the disc volume configuration or to delete tracks, enter YES. Otherwise, enter NO (control skips to Step 39).

29 `LIST VOLUME TABLE?`

To list the disc volume table (as in Step 26), enter YES. Otherwise, enter NO.

Regardless of whether the user requested the listing (by entering YES) or suppressed the listing (by entering NO), at this point in the dialogue, control transfers to Step 30 (if this is a RELOAD) or skips to Step 32 (if this is a COLDSTART, COOLSTART, or UPDATE).

30 `DELETE VOLUME?`

To prepare to delete a volume, enter YES.

To bypass deletion and skip to Step 32, enter NO.

Step No. **Dialogue**

31 `ENTER VOLUME NAME?`

To delete a volume, enter the volume name. (When the name is entered, the question is repeated.)

Otherwise, enter a carriage return.

32 `ADD VOLUME?`

To prepare to add a volume, enter YES.

To bypass addition and skip to Step 34, enter NO.

33 `ENTER VOLUME NAME?`

To add a volume, enter the volume name. (When the name is entered, the question is repeated.)

Otherwise, enter a carriage return.

34 `LIST VOLUME TABLE?`

To list the disc volumes and their currently-assigned logical device numbers (as in Step 26), enter YES.

To suppress this listing, enter NO.

35 `LIST DEFECTIVE TRACKS TABLE?`

To prepare for listing the Defective Tracks Table which resides on any one of the disc volumes, enter YES. To bypass the listing and skip to Step 37, enter NO.

36 `LOGICAL DEVICE #?`

To list the Defective Tracks Table for a particular disc, enter the disc's logical device number. (After this listing, the question is repeated). Otherwise, enter a carriage return.

Step No. **Dialogue**

If the disc is a moving head disc, the following information is printed:

- Logical size of the device (in cylinders)
- Number of alternate tracks available

For each entry in the defective tracks table, the following information is printed:

- Cylinder and head number of the defective track
- Absolute sector number of the first sector of the track (in octal)
- Absolute sector number of the last sector of the track (in octal)
- Track status — may be one of the following:

  (1) SUSPECT — An error has been encountered on this track

  (2) SUSPECT ALT — An error has been encountered on the track to which this track was reassigned

  (3) UNREADABLE ALT — This track was reassigned to another track but the disc driver was unable to read the alternate track assignment

  (4) DELETED — The track is no longer available for use by the system

  (5) REASSIGNED — The track has been reassigned to another track

- The cylinder and head number of the alternate track (if the track status is (2) or (5).)

If the disc is a fixed head disc, the following information is provided in the listing for each track in the table:

- Track number of the defective track
- Absolute sector number of the first sector of the track (in octal)
- Absolute sector number of the last sector of the track (in octal)
- Track status — may be (1) or (4) as described above

Step No. Dialogue

37 **DELETE TRACK?**

To prepare for deleting tracks, enter YES. Otherwise, enter NO to skip to Step 39.

38 **ENTER LDEV, CYLINDER AND HEAD?**

To delete a track on a moving head disc, enter these three parameters, separated by commas: the logical device number of the disc, the cylinder number, and the head number corresponding to the track to be deleted.

To delete a track on a fixed head disc, enter the logical device number and track number, separated by a comma.

Otherwise, enter a carriage return to proceed to Step 39.

If input was entered to delete a track, one of the following messages may result:

ALTERNATE TRACK - CANNOT DELETE

The track is being used as alternate track and cannot be deleted.

IN DIRECTORY - CANNOT DELETE

The track is in the area used by the directory and cannot be deleted since this is not a RELOAD.

IN RESERVED AREA - CANNOT DELETE

The track is in the area reserved for the disc label, bootstrap program, or disk free-space table, and cannot be deleted.

IN SYSTEM AREA - CANNOT DELETE

The track is in the area reserved for the Initiator program and its associated tables, and therefore cannot be deleted.

INVALID CYLINDER NUMBER

The cylinder number is not in the correct range for this moving-head disc.

Step No. Dialogue

```
INVALID HEAD NUMBER
```

The head number is not in the correct range for this moving-head disc.

```
INVALID TRACK NUMBER
```

The track number is not in the correct range for this fixed-head disc.

```
NOT A DISC
```

The device is not a disc.

```
UNINITIALIZED DISC
```

A volume label has not yet been written on the disc.

```
**WARNING** IN DIRECTORY
DELETE?
```

The track is in the area previously used by the directory and this is a RELOAD using the COMPACT or RESTORE option. To delete the track (which will cause the directory to be moved) answer YES. Otherwise, answer NO.

```
**WARNING** IN VIRTUAL MEMORY
DELETE?
```

The track is in the area reserved for the virtual memory. To delete the track, respond with YES; otherwise respond with NO.

Regardless of whether one of these messages was printed, control returns to the beginning of this step.

39 If this system has been configured with the MPE/3000 Logging Facility enabled, the following question is printed:

```
DISABLE LOGGING?
```

To turn off logging until the next cold load, enter YES. Otherwise, respond with NO.

Step No. **Dialogue**

40 If this is not a reload and no tracks have been deleted, the following question is printed:

```
RECOVER LOST DISC SPACE?
```

To recover any disc space that may have been lost because of system failures when temporary files were open, answer YES. Otherwise, respond with NO.

> *Note: For systems with large numbers of files, this may take between 5 and 10 minutes for every 1000 files.*

41 At this point, the Initiator ensures that each disc defined in the configuration has a valid label, that the volume name is defined in the volume table, and that all volumes defined in the volume table are mounted; the Initiator also lists any suspect tracks, suspect alternate tracks, and unreadable alternate tracks and requests the operator to take action on them. As a result of this verification, one or more of the following messages may appear. (All messages require a response from the user. If no messages appear, the Initiator proceeds directly to Step 42.)

(a)
```
INVALID LABEL FOR DEVICE <LDN>
ENTER VOLUME NAME?
```

This indicates that device *ldn* does not contain a valid MPE/3000 volume label. The user must enter a volume name. If the name entered corresponds to that of a volume in the volume table, the logical device number *ldn* is set for that volume. Otherwise, a new entry is made in the volume table. The disc label is then updated.

If device ldn is a moving-head disc, this is followed by the message:

```
LOGICAL PACK SIZE (CYLINDERS)=<SIZE>.?
```

The operator should specify the number of cylinders to be used on this disc; the remainder will be available for alternate track assignments. To retain the default specified by *size* enter a carriage return. Otherwise, enter a value between *minsize* and *maxsize* as defined by the table below:

Subtype	*Size*	*Minsize*	*Maxsize*
2	200	152	203
3	400	304	406

(b)

```
DEVICE <LDN> VOLUME <VOLNAME> NOT DEFINED IN TABLE
ENTER VOLUME NAME?
```

This means that the volume identified by *volname*, with the logical device number *ldn*, does not appear in the volume table. To add this volume, identified by this *volname* and *ldn*, enter a carriage return.

To add this volume under a different volume name, enter the new name. The new name will be entered in the volume table, and the volume will be relabeled with that name.

(c)

```
VOLUME NAME <VOLNAME> ON DEVICE <LDN> ALREADY IN USE
ENTER VOLUME NAME?
```

This means that two volumes have the same name. To change the name of the volume on device *ldn*, enter the new name. If the name corresponds to that of a volume in the volume table, the logical device number *ldn* is set on that volume. Otherwise, a new entry is made in the volume table. The disc label is then updated.

(d)

```
ALL VOLUMES MUST BE MOUNTED
LIST VOLUME TABLE?
```

This message occurs when an entry appears in the volume table but no corresponding volume is mounted on a disc defined in the configuration. To list the volume table (as in Step 26), respond with YES. Otherwise, respond with NO. In either case, control then returns to Step 7.

Step No. Dialogue

(e)
```
SUSPECT          )
SUSPECT ALT      } TRK LDEV#<LDN> CYL:<CYL> HEAD:<HEAD>
UNREADABLE ALT   ) SECTORS %<FSECT>-%<LSECT>
```

This message, applying to a moving-head disc, indicates that an entry in the Defective Tracks Table requires action. (The variable parameters and other related information are described under Message (f) below.

(f)
```
SUSPECT TRK LDEV #<LDN> TRACK:<TRACK> SECTORS %<FSECT>-%<LSECT>
```

This message, applying to a fixed-head disc, indicates that an entry in the Defective Tracks Table requires action.

The following information applies to both Messages (e) and (f) above. A *suspect track* is one on which an error has been detected while the system was running. A *suspect alternate track* is one that has been previously reassigned and an error has been detected on the alternate track. An *unreadable alternate track* is one that has been reassigned but the disc driver was unable to read the address of the reassigned track while attempting a transfer. In Messages (e) and (f), *ldn* gives the logical device number of the disc where the error occurred; *track* gives the track in error for fixed head discs; *cyl* and *head* give the cylinder and head number of the bad track for moving head discs; and *fsect* and *lsect* give the absolute sector number of the first and last sectors of the track.

Immediately following Message (e) or (f), one of the Messages (g) through (q) may appear; they require one of the following replies, or YES or NO as indicated:

(1) DELETE To remove the track from the space available for use by the system

(2) REASSIGN To reassign the space on the bad tracks to one of the available alternate tracks

(3) RECOVER To remove the entry from the defective tracks table, ignoring the error

(4) (A *carriage return*) To ignore the error. The message will be repeated the next time the system is cold-loaded.

Step No. **Dialogue**

(g)

```
DELETE OR REASSIGN?
```

The track was flagged as defective previously, and resides on a disc that had an invalid label. Only Responses (1) and (2) are valid.

(h)

```
**WARNING** IN ALTERNATE AREA
DELETE?
```

The track was flagged as defective previously, and resides on a disc that had an invalid label; it is located in the area reserved for alternate tracks. The only valid response is *YES*, since tracks in the alternate area cannot be reassigned (reassigned tracks cannot be "chained").

(i)

```
**WARNING** IN ALTERNATE AREA
DELETE OR RECOVER?
```

A suspect track is located in the area reserved for alternate track assignment. Valid Responses are (1), (3) and (4).

(j)

```
FLAGGED TRACK IN RESERVED AREA - MUST REINITIALIZE PACK
```

A track flagged as defective is located in the area reserved for the disc label and Disc Free-Space Table on a volume having an invalid label. The system halts at this point. The operator must either rerun the diagnostic on the pack, removing the defective track flag from this track, or he must mount a new pack. Following this, he should restart the cold-load sequence.

(k)

```
**WARNING** IN RESERVED AREA
RECOVER?
```

A suspect track is located in the area reserved for the disc label and Disc Free-Space Table. Respond with YES to remove the entry from the Defective Tracks Table, or NO to ignore it (same action as in Response (4), above).

Step No. Dialogue

(l)

```
**WARNING** IN SYSTEM AREA
RECOVER?
```

A suspect track is located in the area used for the Initiator program and its associated tables. Respond with YES to remove the entry, or NO to ignore it.

(m)

```
**WARNING** IN DIRECTORY
RECOVER?
```

A suspect track is located in the area used by the system for the file directory, and this is not a RELOAD. Respond with YES to remove the entry or NO to ignore it.

(n)

```
 **WARNING** IN DIRECTORY
{DELETE, REASSIGN OR RECOVER?}
{DELETE OR RECOVER?          }
```

A suspect track is located in the area previously assigned to the directory and this load is a RELOAD using the COMPACT or RESTORE options. The first form of the question is used for moving head discs and the second for fixed head discs. All Responses are valid except for (2) in answer to the second form of the question.

(o)

```
 **WARNING** IN VIRTUAL MEMORY
{DELETE, REASSIGN OR RECOVER?}
{DELETE OR RECOVER?          }
```

A suspect track is located in the area used for the virtual memory. If this load is not a RELOAD, the second form of the question will be used and Responses (1), (3) and (4) are valid. If it is a RELOAD using the COMPACT or RESTORE options, the first form of the question will be used for moving head discs and the second form for fixed head discs. All Responses are valid except for (2) in answer to the second form of the question.

(p)

```
DELETE OR RECOVER?
```

The suspect track is located in no special area on a fixed head disc. Responses (1), (3) and (4) are valid.

Step No. **Dialogue**

(q)
```
DELETE, REASSIGN OR RECOVER?
```

The suspect track is located in no special area of a moving head disc. All Responses are valid.

42 If any changes have been made to the defective tracks table in Step 41, the following question will be printed:

```
LIST DEFECTIVE TRACKS TABLE?
```

To prepare for listing the defective tracks table of any disc, respond with YES. Otherwise, respond with NO to skip to Step 44.

43
```
LOGICAL DEVICE #?
```

Enter the logical device number of the disc for which the defective tracks table listing is desired, (as in Step 36). This question is repeated until a carriage return is entered, signifying that no further listings are desired.

44 If any changes have been made to the Volume Table in Step 41, the following question is printed:

```
LIST VOLUME TABLE?
```

Respond with YES to obtain a listing of the volume table (as described in Step 26); otherwise, enter NO to bypass the listing.

45 If this load is not a RELOAD and the size of virtual memory on the disc differs from the configured size of virtual memory, the following message will be printed:

```
**WARNING** VIRTUAL MEMORY SIZE ONLY CHANGED ON RELOAD
```

This message will often occur when reconfiguring the system for a different main-memory size, since the default value for the size of the virtual memory differs for different main-memory sizes.

Step No. Dialogue

46 If this load is not a RELOAD and the size of the directory on disc differs from the configured size, the following message will be printed:

```
**WARNING** DIRECTORY SIZE ONLY CHANGED ON RELOAD
```

This often occurs for the same reason described in Step 45.

47 If this load is not a RELOAD and the size of the RIN table or the global area of the RIN table differs from the configured size, the following message will be printed:

```
**WARNING** RIN TABLE ONLY CHANGED ON RELOAD
```

This often occurs for the same reason noted in Step 45.

48 If this load is not a RELOAD and any tracks were deleted or reassigned, the following list may be printed:

```
FOLLOWING FILES PURGED - DISC ERROR
```

filename.groupname.accountname

•
•
•

These files resided wholly or partially on tracks that were deleted or reassigned and were therefore purged from the directory. The space used by the files, except for the area of the defective track, was returned to the disc free-space list.

49 If this is a COLDSTART, COOLSTART, UPDATE, or RELOAD with no user files, control proceeds to Step 50. Otherwise, the user files are read from the tape and written to the disc. As a result, one or more of the following messages may be printed:

(a) `MOUNT REEL #<REELNUM>`

The next reel of the set, *reelnum*, should be mounted and placed on line.

(b) `TAPE NOT A MEMBER OF THIS SET`

The tape mounted in response to Message (a) was not a member of the tape set.

(c) `WRONG REEL`

Reel *reelnum*, specified in Message (a), was not mounted. Rather, another reel in the same tape set was mounted.

Step No. Dialogue

(d) NOT A RELOAD TAPE

The tape mounted in response to Message (a) is not a tape generated by :SYSDUMP or :STORE

(e) NOT ALL FILES FOUND: ANOTHER TAPE SET AVAILABLE?

Not all of the files in the directory when the system was dumped have been found on the tape. If another set of tapes is available, respond with YES, mount the first reel of the set, and place it on line. Otherwise, respond with NO.

(f) WRONG TAPE SET - MUST HAVE EARLIER DATE

The tape mounted in response to Message (e) has a date later than the previous tape set processed. The operator should find the first reel of the correct set, mount it, and place it on line.

(g) FOLLOWING FILES PURGED - NOT FOUND

```
filename.groupname.accountname
      .
      .
      .
      .
```

If the response to Message (e) was NO, a list of files that were in the directory but were not found on the tapes (and were therefore purged) is printed.

(h) FOLLOWING FILES PURGED - INSUFFICIENT DISC SPACE

```
filename.groupname.accountname
      .
      .
      .
      .
```

If there was insufficient disc space to reload all the files, those files that had to be purged are listed.

Step No. Dialogue

50 <u>DATE?</u>

Enter the current date in the following format:

```
mm/dm/yr
```

where

mm = Two digits representing the month
dm = Two digits representing the day of the month
yr = The last two digits of the year.

51 <u>TIME?</u>

Enter the current time-of-day in the following format:

```
hh:mm
```

where

hh = Two digits indicating the hour (on a 24-hour basis).
mm = Two digits indicating the minute of the hour.

The Initiator program now terminates, transferring control to MPE/3000. The system is ready for use.

Load Map

The load map requested in Step 4 of the Initiator/User Dialogue appears as shown in figure 5-2. This map shows the correspondence between MPE/3000 code segments and programs, and code segment table (CST) entries. In the first column of this listing, the CST number (in octal) is shown. In the second column, the System Segmented Library (SL) segment name or program name is presented. (SL segment names are followed by a parenthesized value in the third column; program file names are not.) In the third column, the parenthesized number indicates the logical segment number of the segment within the system library (identified as SL.PUB.SYS).

```
 20  FILESYS1 (0)
 21  FILESYS2 (1)
 22  FILESYS3 (2)
 23  FILESYS4 (3)
 24  FILESYS5 (4)
 25  FILESYS6 (5)
 26  FILESYS6A (6)
 27  FILESYS7 (7)
 30  CISUBS (10)
 31  CIORGMAN (11)
 32  CIINIT (12)
 33  CIFILEM (13)
 34  CIFILEB (14)
 35  CILISTF (15)
 36  CIMISC (16)
 37  CIERR (17)
 40  CXSTOREST (20)
 41  STORE (21)
 42  RESTORE (22)
 43  DIRC (23)
 44  ALLOCATE (24)
 45  DISKSPC (25)
 46  MMCORER (26)
 47  MMDISKR (27)
 50  ABORTRAP (30)
 51  MESSAGE (31)
 52  CROUTINE (32)
 53  IOUTILTY (33)
 54  TTYINT (34)
 55  PCREATE (35)
 56  MORGUE (36)
 57  PROCMAIL (37)
 60  PINT (40)
 61  DATASEG (41)
 62  CHECKER (43)
 63  UTILITY (44)
 64  SEGUTIL (45)
 65  LOADER1 (46)
 66  RINS (47)
 67  JOBTABLE (50)
 70  DEBUG (51)
 71  SYSDEBUG (52)
 72  SYSDSPLY (53)
 73  IOPM (42)
 74  TRACE0' (104)
 75  CLIB'03 (57)
 76  CLIB'07 (63)
 77  TRACE1' (105)
100  ININ
101  DISPATCH
102  MAPP
103  IOFDISK0
104  IOMDISK0
105  IOCLTTY0
106  IOTAPE0
107  IOCDRD0
110  IOLPRT0
111  IOTERM0
112  IOPLOT0
```

Figure 5-2. Load Map

Cold Load Error Messages

One of the following error messages may appear while cold loading the system. Following the printing of the message, the machine halts.

BAD DISK ADDRESS

An address greater than the available number of sectors on the disc was passed to the disc driver. This usually indicates an error internal to the system.

BAD FILE ADDRESS

An attempt was made to write outside the range of one of the system files.
A RELOAD should be attempted.

COLD LOAD TAPE READ ERROR

A tape-read error was detected during the cold load operation.

DIRECTORY ERROR A=<A>, B=<B>

An error occurred while accessing the directory. The error can be interpreted as follows:

A=1	Duplicate name		
A=2	Non-existent	B=0	file
		B=1	group
		B=2	account
		B=3	user
A=4	No index room		
A=5	Too many directory entries		
A=6	Out of directory space		

DISC SPACE ERROR

A conflict exists between the disc free-space map and the space used as defined in the directory. A RELOAD should be attempted.

DISK DRIVER DOES NOT EXIST

A transfer has been attempted to a disc with a type or subtype not known to the system. This usually indicates an error internal to the system.

DISK $\left\{\begin{array}{l}\text{READ}\\ \text{WRITE}\\ \text{SEEK}\end{array}\right\}$ ERR ON LDEV #<LDEV> STATUS=%<STATUS> ADDR=%<ADDR> WORDS=<WORDS>

A disc error has occurred on the specified logical device. The operator should cold-load again, as he will be prompted to take action on the bad track.

EOF

An attempt has been made to read past the end of one of the system files. A RELOAD should be attempted.

FILE <NAME>.PUB.SYS NOT ON DISC

The specified file was needed but was not found in the directory.

IMPROPER TAPE FORMAT

The information on a tape used for RELOAD does not agree with the format of tapes produced by the :STORE and :SYSDUMP commands.

IOP ERROR

An *impossible* status indication was returned by the tape controller, indicating a failure in the controller or the input/output processor.

MOUNT CORRECT VOLUMES OR RELOAD

On a COOLSTART, COLDSTART or UPDATE, not all of the previously-defined volumes were found. The operator should either mount the correct volumes and start the COOLSTART, COLDSTART or UPDATE over, or he should RELOAD.

OUT OF BOOTSTRAP DISC SPACE

The 20 sectors of the system disc allocated to the bootstrap program has been exceeded; this is an error internal to the system.

OUT OF CORE MEMORY

The amount of space needed to build the main-memory resident portion of the system, and contain one segment of the Initiator program and its associated tables, has exceeded the available core memory.

OUT OF SYSTEM DISC SPACE

The virtual memory, directory, and system file disc space required exceeds that available on the system disc.

PREVIOUS RELOAD ABORTED; MUST RELOAD

The last cold-load was a RELOAD that was aborted. Therefore, this cold-load must be a RELOAD.

PREVIOUS TAPE COLD LOAD ABORTED; MUST COLD LOAD FROM TAPE

The last cold-load was a COLDSTART or UPDATE that was aborted. Therefore, this cold-load must be a COLDSTART, UPDATE or RELOAD.

READING BLANK TAPE

This error, reported by the magnetic tape controller, implies either a bad tape or a bad tape controller.

TAPE I/O CMD REJECTED

This error, reported by the magnetic tape controller, implies that there is a bad tape controller.

TAPE PARITY ERROR

A parity error was detected while reading the magnetic tape.

TAPE TRANSFER ERROR

This error, reported by the magnetic tape controller, implies either a bad tape or a bad tape controller.

TAPE UNIT WENT NOT READY

This error, reported by the magnetic tape controller, implies either that the tape controller is bad or the operator switched the tape unit off-line during an operation.

TIMING ERROR

This error, reported by the magnetic tape controller, implies a bad tape controller.

VOLUME TABLE DESTROYED; MUST RELOAD

The Volume Table maintained on the system disc has been over-written; the system must be reloaded.

SHUTDOWN

The user can shut the system down by following these procedures.

1. Press the CONSOLE INTERRUPT Button on the Computer Control Panel. In response, MPE/3000 prints an equals sign (=) on the MPE/3000 Console.

2. Enter the SHUTDOWN command after the equals sign (and terminate the command by pressing the carriage return key) as follows:

```
=  SHUTDOWN
```

The =SHUTDOWN command shuts the system down in an orderly manner, terminating all jobs and sessions currently in progress. The message *SHUT* is printed, and the central processor halts with %030377 in the current instruction register (CIR). (The machine *must* be allowed to run until these events occur.) The system can be restarted at any time, as directed in the preceding pages.

SECTION VI
Organizational Management

A user with the MPE/3000 System Manager Capability can create, modify, and delete accounts. With each account, the System Manager User also creates a user with the Account Manager Capability and a Public Group (named PUB). The Account Manager User can then create, modify, and delete groups and users within his account. (The Public Group is a group whose file library is normally accessible for reading and program execution to all users within the account.)

TYPICAL ACCOUNTS

To illustrate the various ways in which accounts, groups, and users can be defined and organized at an HP 3000 Computer site, several examples are presented. Notice that these are only examples, and are not meant to imply formal organizational rules.

Large Multi-Division Company

Within a system used to provide interactive terminals in a large, multi-division corporation, a System Manager User could assign a set of accounts to each division or corporate staff office according to its functional needs (Figure 6-1). For example, he could assign four accounts to Division A — one for each of its departments: Engineering, Production, Marketing, and Finance. For the Engineering Department account (detailed in Figure 6-1), an Account Manager User has defined three users who can access the system. Each has a private group (assigned as his home group) where he stores his private programs and files. In addition, these engineers all can access two special groups that contain programs and data files for current design projects. When an engineer uses the system to design work on Project 1, he logs onto that group, and his central processor time, connect-time, and disc-space used for permanent files is charged against that group (project). The engineers can also access a group defined for work on schedules, budgets, and other administrative tasks, and a public group (with no password required at log-on time) for storage of general-purpose utility programs used by all.

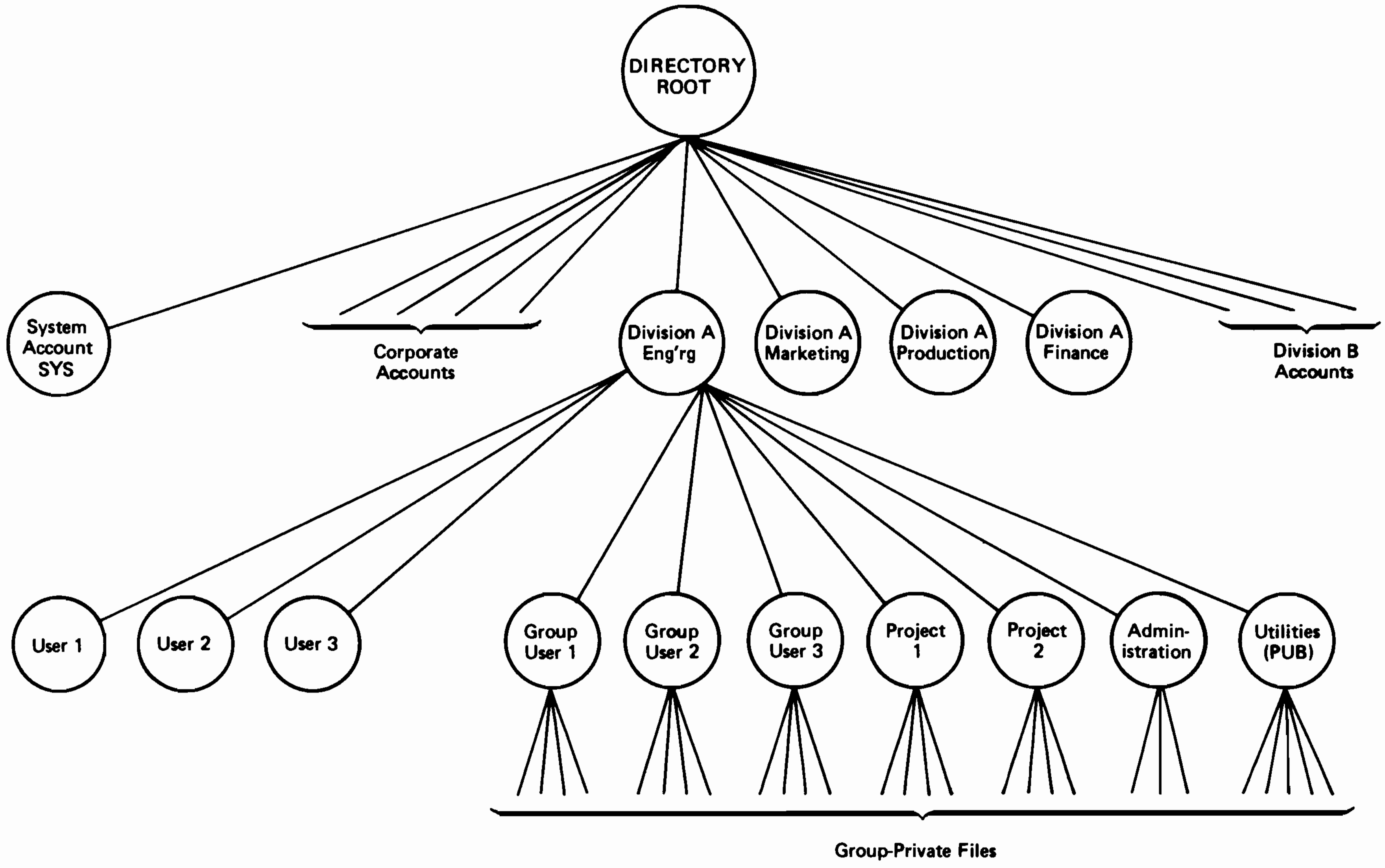


Figure 6-1. Account/Group/User Structure in a Large Corporation

In-House Service Bureau

An in-house service bureau, providing both batch and interactive capabilities, serves different users within the same company. The manager of the Data Center, having the System Manager Capability, creates an account for each department (Engineering, Marketing, Accounting, and other departments) in the company. For each account an Account-Manager User assigns as users all programmers in his department, and assigns them all Standard Capabilities plus the following Optional Capabilities: Process-Handling, Data-Segment Handling, and Multiple RINS. The Account Manager User also defines the groups for the account.

As an example of a typical account, the account for the Engineering Department defines all programmers in that department as users. This account contains the following groups:

- Public Group, containing utility programs, for which all users of the account have execute or read access only. (Types of access are discussed in Appendix I of this manual, and in *HP 3000 Multiprogramming Executive Operating System (03000-90005).*)

- Data-Collection Group, for which all users have read, write (append only), save, and file-locking access.

- Common Group, used for miscellaneous department overhead. Unrestricted access; no passwords necessary.

- Project Groups (one per project). Access restricted to project members.

- Private Groups (one for each programmer with private password).

The MPE/3000 Accounting System would be used, to some extent, for internal bookkeeping. This system monitors permanent file space and central-processor time used by job/sessions for accounts and groups, and interactive terminal connect-time used by sessions for accounts and groups.

Commercial Service Bureau

A commercial service bureau, also providing batch and interactive capabilities, offers its services to various outside companies (customers). An account is established for each customer. Users are given Standard Capabilities but no Optional Capabilities. The group organization is similar to that for the in-house service bureau described above. The MPE/3000 Accounting System is used more fully in the commercial service bureau, however.

Scientific Research Center

In a research center where scientific programmers are using the system for batch and interactive applications, the following accounts are available:

- System Account, used by all users.

- Data-Processing account, containing groups allocated as required, used by all research users. For this account, all users have the Standard Capabilities and the following Optional Capabilities: Process-Handling, Data-Segment Handling, and Multiple RIN'S.

- Special Accounts, used by agencies such as the Personnel and Accounting Offices. Access is restricted to members of these agencies.

In this system, accounting is not critical, and the MPE/3000 Accounting System could be used minimally.

DEFINING ACCOUNTS

When a user with System Manager Capability accesses MPE/3000 to perform various functions, he logs on to the system in the same way that any other user does. He must, however, present the appropriate identification and password that identifies himself as a user with System Manager capability. Then, he can enter the commands for managing accounts described below.

The first System Manager User is designated by HP on the system tape supplied the customer. He initially accesses the system under the user name MANAGER, the group name PUB, and the account name SYS. (The initial passwords are null.)

Creating Accounts

To create a new account and an associated Account Manager User and Public Group (named PUB), the System Manager User enters the :NEWACCT command. (The new Account Manager User can then redefine his own attributes, the Public Group, and new users and groups as he desires.) The Public Group is initially assigned a null password and the same capability-class attributes, permanent file space limit, central processor time limit, and connect time limit as the account. Its initial security grants reading and program-execute access to all users who can pass the account's security, and appending, writing, locking, and saving access to account librarian and group users only. (These access provisions are defined syntactically as (R, X: ANY; A, W, L, S: AL, GU). This notation is used throughout the remainder of this manual; it is explained fully in Appendix I of this manual and in the discussion of MPE/3000 file security appearing in *HP 3000 Multiprogramming Executive Operating System (03000-90005).*)

The format of the :NEWACCT command follows.

NOTE: *All names, such as acctname, mgrname, and password, are composed of up to eight alphanumeric characters, beginning with a letter.*

```
:NEWACCT   acctname, mgrname
     [;PASS = [password] ]
     [;FILES = [filespace] ]
     [;CPU = [cpu] ]
     [;CONNECT = [connect] ]
     [;CAP = [capabilitylist] ]
     [;ACCESS = [fileaccess] ]
     [;MAXPRI = [subqueuename] ]
     [;LOCATTR = [localattribute] ]
```

acctname — The name assigned to the new account. (Required parameter.)

mgrname — The name of the account manager user; he is always the first user created under the account. He receives the following attributes:

User password:	Null password.
User capability list:	Same as Account's capabilities.
Maximum scheduling priority:	Same as Account's maximum priority.
Local attribute:	Same as Account's local attribute.
Home group:	PUB

(The attributes of an Account Manager User may be changed subsequently, through the :ALTUSER command; in no case, however, is this user granted *effective* attributes greater than those of the account.) (Required parameter.)

password — The account password. (This password is used for verifying log-on access only.) If omitted, no password is assigned. (Optional parameter.)

filespace — The disc storage limit, in sectors, for the permanent files of the account. The maximum value permitted is 2 billion sectors. If omitted, no limit on permanent file space is assigned. (Optional parameter.)

cpu — The limit on total central processor time, in seconds, for the account. (This limit is checked against the actual time accumulated only when a job/session is initiated — thus, the limit never causes a job/session to abort once it is in progress.) The maximum value permitted is approximately 2 billion (2,147,483,647 = ($2^{31}-1$)) seconds. If omitted, no limit on central processor time is assigned. (Optional parameter.)

connect — The limit on total session connect-time, in minutes, allowed the account. Like the *cpu* limit, this limit is checked against actual usage only at log-on time. The maximum value allowed is 2 billion minutes. If omitted, no limit on connect-time is assigned. (Optional parameter.)

capabilitylist The list of capabilities, mutually separated by commas, permitted this account. Each capability (described in Section II) is denoted by a two-letter mnemonic, as follows:

User Attributes:

System Manager	= SM	
Account Manager	= AM	(Assigned automatically to every account.)
Account Librarian	= AL	
Group Librarian	= GL	
Diagnostician	= DI	
System Supervisor	= OP	

File-Access Attributes:

Permanent Files	= SF
Access of non-sharable I/O devices	= ND

Capability-Class Attributes:

Process-Handling	= PH	
Extra Data Segments	= DS	
Multiple RINS	= MR	
Privileged Mode	= PM	
Interactive Access	= IA	At least one of these must be specified.
Local Batch Access	= BA	

If the *capabilitylist* parameter is omitted, the following capabilities are assigned by default:

AM,AL,GL,SF,IA,BA

(Optional parameter.)

fileaccess The restrictions on file access pertinent to this account, entered in the same format as the *modelist:userlist* parameter in the :ALTSEC Command described in *HP 3000 Multiprogramming Executive Operating System (03000-90005).* The default value for all accounts is (R,A,W,L,X: AC). (Optional parameter.)

subqueuename The name of the subqueue of highest priority that can be requested by any process of any job in the account, specified as "*x*S," where *x* is A, B, C, D, or E. If none is specified, the CS subqueue is assigned. (Optional parameter.)

localattribute The local attribute of the user, as defined at the installation site. If omitted, 0 is assigned. This is a double-word of arbitrary meaning which might be used to further classify users. While it is not involved in standard MPE/3000 security provisions, it is available to process through the WHO intrinsic for use in the programmer's own security provisions. (Optional parameter.)

In the :NEWACCT command, the default parameters noted are assigned:

1. When a keyword is included but its corresponding keyword parameter is omitted (as in ;PASS=).

2. When an entire keyword parameter grouping (such as ;PASS= password) is omitted.

EXAMPLE:

Suppose that a System Manager wants to create an account with the following characteristics:

Account name = ACI
Account password = PCI
Manager's name = SMYTHE

(All other parameters are assigned by default.)

To accomplish this, the System Manager User could enter:

```
:NEWACCT ACI,SMYTHE; PASS=PCI
```

Altering Accounts

A System Manager User can change the attributes of an existing account by entering the :ALTACCT command.

```
:ALTACCT    acctname
      [;PASS = [password]]
      [;FILES = [filespace]]
      [;CPU = [cpu]]
      [;CONNECT = [connect]]
      [;CAP = [capabilitylist]]
      [;ACCESS = [fileaccess]]
      [;MAXPRI = [subqueuename]]
      [;LOCATTR = [localattribute]]
```

acctname The name of the account to be altered. (Required parameter.)

password
filespace
cpu
connect
capabilitylist
fileaccess
subqueuename
localattribute

New values for the corresponding parameters described under the :NEWACCT command. The maximum limits for these parameters discussed under :NEWACCT also apply to :ALTACCT. The *filespace* limit cannot be less than the number of sectors currently allocated for the account. When altering the *capabilitylist* for the SYS account, the SM capability cannot be removed. If *acctname* is SYS, and the *fileaccess* parameter is omitted the following default security is is assigned: (R, X: ANY; A, W, L: AC). Other default values are as described under :NEWACCT. (Optional parameters.)

When an entire keyword parameter group is omitted from the :ALTACCT command, that parameter remains unchanged for the account. When a keyword is included but the corresponding parameter is omitted (as in ;PASS=), the default value is assigned. Thus, when changing one capability in a *capabilitylist* presently containing several non-default values, the user must re-specify the *entire* new *capabilitylist* — not just the changed parameter.

Any value changed through the :ALTACCT command becomes effective the next time MPE/3000 is requested to check this value; if an attribute is taken away from an account, users of the account currently running with that attribute will retain it until again logging on.

EXAMPLE:

To change an account named AC2 so that its password is GLOBALX and its filespace is limited to 50,000 sectors, a System Manager User could enter:

```
:ALTACCT AC2; PASS=GLOBALX; FILES=50000
```

Listing Account Attributes

To list the attributes (previously specified in the :NEWACCT or :ALTACCT commands) for one account, or for all accounts in the system, a System Manager or Account Manager User uses this command:

```
:LISTACCT ⎡@       ⎤ [,listfile]
          ⎣acctname⎦
```

@ An indication that the attributes for all accounts are to be listed. Restricted to System Manager Users. (Optional parameter.)

acctname The name of the account whose attributes are to be listed. Can be specified by System Manager Users. *Must* be specified by Account Manager Users who do not *also* have System Manager Capability; furthermore, such users can only specify their own account. (Optional parameter for users with System Manager Capability. Required parameter for users with Account Manager Capability only.)

listfile The actual designator of the file on which the attribute listing is produced. The options assigned this file when it is opened (through the FOPEN intrinsic) are:

NEW, ASCII, VARIABLE, CCTL, OUTPUT (RETAIN DATA), EXCL

The file is closed (through the FCLOSE intrinsic) with SAVE disposition. If this parameter is omitted, the job/session list device ($STDLIST) is assigned. (Optional parameter.)

If neither @ nor *acctname* is specified by a System Manager User, all accounts (@) are listed.

EXAMPLE:

To list the attributes of his own account, named ACCTNO23, the Account Manager User enters:

```
:LISTACCT ACCTNO23
```

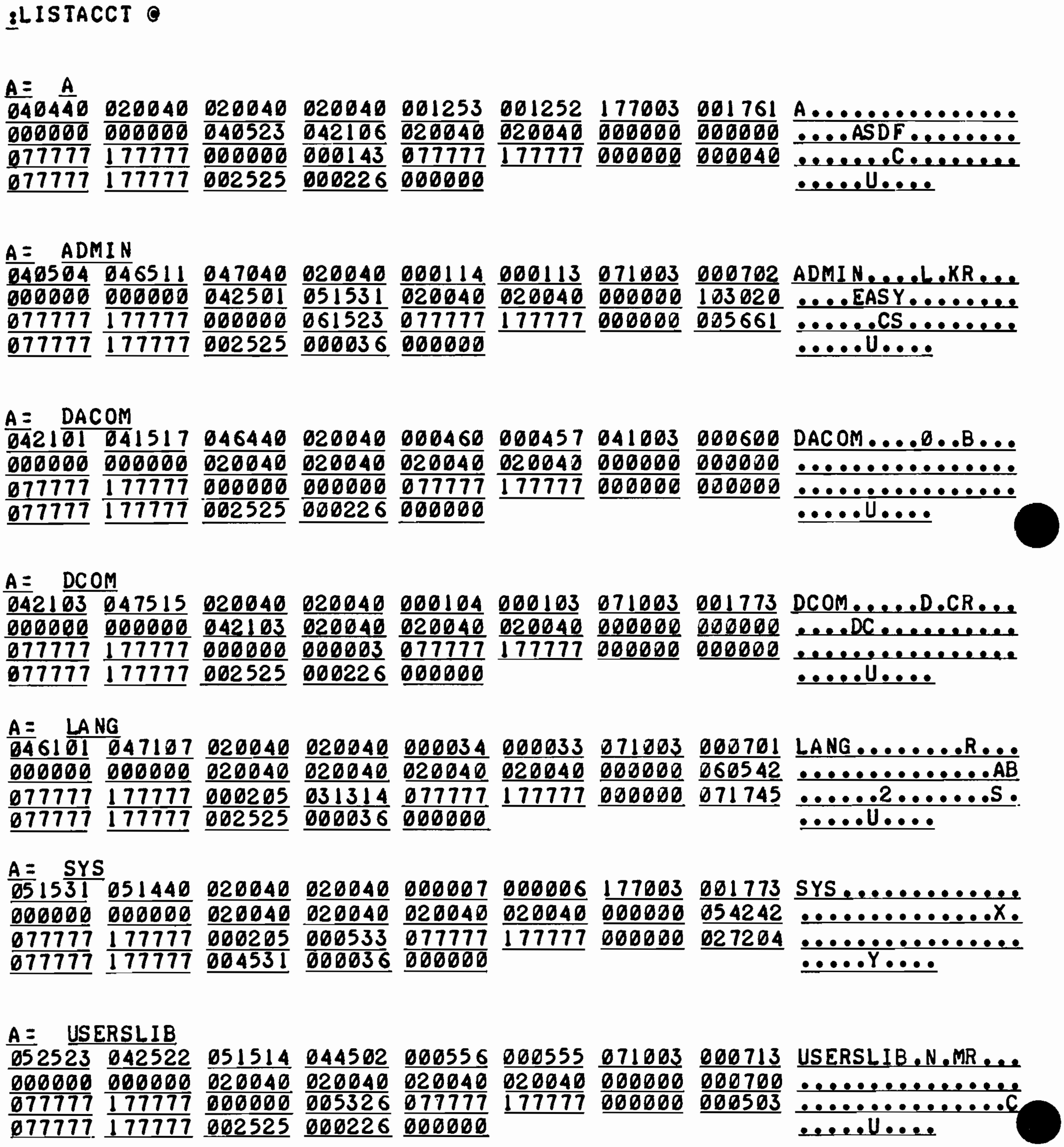
The listing appears as an octal dump of the account entry or entries. Each entry is headed by the character *A* (for account), an equals sign, and the account name. The user information in the entry is decoded as shown below; both user and system information is decoded as shown in Appendix J.

Words	Content
0-3	Account name.
6-7	Capability (in same format returned by WHO intrinsic, described in *Multiprogramming Executive Operating System*).
8-9	Local attributes.
10-13	Password
14-15	Permanent file space usage count (in sectors).
16-17	Permanent file space limit (in sectors).
18-19	Central processor time usage count (in seconds).
20-21	Central processor time limit (in seconds).
22-23	Connect-time count (in minutes).
24-25	Connect-time limit (in minutes).
26	Purge and account-security flags (see Appendix K).
27	Maximum job/session priority (numerical).

A translation of those bytes that contain alphanumeric ASCII characters appears to the right of each line in the octal dump.

EXAMPLE:

The following :LISTACCT command produces the listing that appears below it.



```
:LISTACCT @


A=  A
Ø4Ø44Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØ1253  ØØ1252  1770Ø3  ØØ1761  A...............
ØØØØØØ  ØØØØØØ  Ø40523  Ø42106  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØØØØØ  ØØØØØØ  ....ASDF........
Ø77777  177777  ØØØØØØ  ØØØ143  Ø77777  177777  ØØØØØØ  ØØØØ4Ø  .......C........
Ø77777  177777  ØØ2525  ØØØ226  ØØØØØØ                          .....U....


A=  ADMIN
Ø4Ø5Ø4  Ø46511  Ø47Ø4Ø  Ø2ØØ4Ø  ØØØ114  ØØØ113  Ø71ØØ3  ØØØ7Ø2  ADMIN....L.KR...
ØØØØØØ  ØØØØØØ  Ø425Ø1  Ø51531  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØØØØØ  1Ø3Ø2Ø  ....EASY........
Ø77777  177777  ØØØØØØ  Ø61523  Ø77777  177777  ØØØØØØ  ØØ5661  ......CS........
Ø77777  177777  ØØ2525  ØØØØ36  ØØØØØØ                          .....U....


A=  DACOM
Ø421Ø1  Ø41517  Ø4644Ø  Ø2ØØ4Ø  ØØØ46Ø  ØØØ457  Ø41ØØ3  ØØØ6ØØ  DACOM....Ø..B...
ØØØØØØ  ØØØØØØ  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØØØØØ  ØØØØØØ  ................
Ø77777  177777  ØØØØØØ  ØØØØØØ  Ø77777  177777  ØØØØØØ  ØØØØØØ  ................
Ø77777  177777  ØØ2525  ØØØ226  ØØØØØØ                          .....U....


A=  DCOM
Ø421Ø3  Ø47515  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØØ1Ø4  ØØØ1Ø3  Ø71ØØ3  ØØ1773  DCOM.....D.CR...
ØØØØØØ  ØØØØØØ  Ø421Ø3  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØØØØØ  ØØØØØØ  ....DC..........
Ø77777  177777  ØØØØØØ  ØØØØØ3  Ø77777  177777  ØØØØØØ  ØØØØØØ  ................
Ø77777  177777  ØØ2525  ØØØ226  ØØØØØØ                          .....U....

A=  LANG
Ø461Ø1  Ø471Ø7  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØØØ34  ØØØØ33  Ø71ØØ3  ØØØ7Ø1  LANG........R...
ØØØØØØ  ØØØØØØ  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØØØØØ  Ø6Ø542  ..............AB
Ø77777  177777  ØØØ2Ø5  Ø31314  Ø77777  177777  ØØØØØØ  Ø71745  ......2.......S.
Ø77777  177777  ØØ2525  ØØØØ36  ØØØØØØ                          .....U....

A=  SYS
Ø51531  Ø5144Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØØØØ7  ØØØØØ6  177ØØ3  ØØ1773  SYS.............
ØØØØØØ  ØØØØØØ  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØØØØØ  Ø54242  ..............X.
Ø77777  177777  ØØØ2Ø5  ØØØ533  Ø77777  177777  ØØØØØØ  Ø272Ø4  ................
Ø77777  177777  ØØ4531  ØØØØ36  ØØØØØØ                          .....Y....


A=  USERSLIB
Ø52523  Ø42522  Ø51514  Ø445Ø2  ØØØ556  ØØØ555  Ø71ØØ3  ØØØ713  USERSLIB.N.MR...
ØØØØØØ  ØØØØØØ  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  Ø2ØØ4Ø  ØØØØØØ  ØØØ7ØØ  ................
Ø77777  177777  ØØØØØØ  ØØ5326  Ø77777  177777  ØØØØØØ  ØØØ5Ø3  ...............C
Ø77777  177777  ØØ2525  ØØØ226  ØØØØØØ                          .....U....
```

Deleting Accounts

To remove an account (and its groups (files) and users) from the system, a System Manager User enters:

```
:PURGEACCT acctname
```

acctname The name of the account to be deleted. (Required parameter.)

When the :PURGEACCT command is entered in a session, MPE/3000 prints a verification request. This aids the user in making sure he does not accidentally delete the wrong account. The format of the verification request is

```
ACCT accountname TO BE PURGED?
```

The user responds with YES or NO.

(No verification is made when the :PURGEACCT command is entered in a job.)

If no users are logged-on under the account, and none of its groups/files are in use (as defined in the note below), :PURGEACCT removes the entire account. Otherwise, :PURGEACCT removes every user not currently logged on and every group/file not in use. This could leave the account (containing residual elements only now) still in the system; in such a case, if all users were purged, the account would exist but would be inaccessible for log-on. In such cases, to completely purge the account, the user must re-issue the :PURGEACCT command when the account is inactive.

The system account, SYS, cannot be purged.

NOTE: *A* file *is in use if it is currently opened (FOPEN intrinsic), contains a program being executed (:RUN command or CREATE intrinsic), or is being stored (:STORE command) or restored (:RESTORE command).*

A group *is in use if any of its files are in use, if it is being used as a home group, or if a :LISTF@ command is applied to its files.*

An account *is in use if any user is logged-on under it; any of its groups are in use; or any of the following commands are applied to it:*

```
:LISTGROUP @
:LISTUSER @
:REPORT @
:RESETACCT
```

EXAMPLE:

When a System Manager User removes an account named ACCTXYZ from the system during a session, the following dialogue takes place:

```
:PURGEACCT ACCTXYZ
ACCT ACCTXYZ TO BE PURGED? YES
```

DEFINING GROUPS AND USERS

Once an account has been defined and someone has been assigned Account Manager Capability for that account, that person can manage groups and users within the account. The Account Manager User exercises this control through the commands noted below.

Creating Groups

An Account Manager User creates a new group within his account by issuing the :NEWGROUP command. In doing so, he also optionally assigns a password for this group. Thereafter, any user who knows the password can log-on using that group. (The password is not needed to access the files directly.) Additionally, the group can be assigned as a homegroup for any user defined (in the :NEWUSER command, described later). The :NEWGROUP command format is

```
:NEWGROUP groupname
      [;PASS = [password] ]
      [;CAP = [capabilitylist] ]
      [;FILES = [filespace] ]
      [;CPU = [cpu] ]
      [;CONNECT = [connect] ]
      [;ACCESS = [fileaccess] ]
```

groupname The name of the new group, composed of up to eight alphanumeric characters, beginning with a letter. (Required parameter.)

password The same meaning and default value as the corresponding parameter in the :NEWACCT command, but applicable to the group. (Optional parameter.)

capabilitylist A list of *capability-class attributes*, consisting of any or all of the following: IA, BA, PM, MR, DS, or PH (as specified in the discussion of the :NEWACCT command). This list imposes a limit on program files belonging to the group. A capability cannot be defined if it is not presently possessed by the group's account. However, if the account is later altered, the group could be left with excessive capabilities. (*System Manager Users should be aware of this when changing an account's* capability list.)

The default assigned is IA, BA. (Optional parameter.)

filespace
cpu
connect

The same meanings and maximum values as the corresponding parameter in the :NEWACCT command, but applicable to the group. A group's *filespace*, *cpu*, or *connect* limits cannot be specified as greater than the corresponding limits currently defined for the group's account. (The default values imply *no* limits.) However, an account's *filespace*, *cpu*, or *connect* limits can later be changed so that some of its groups are left with limits that exceed the new account limits.

fileaccess

The same meaning as the corresponding parameter in the :NEWACCT command, but applicable to the group. If omitted, the specification assigned by default is:

For Public Group (PUB):	(R,X:ANY; A,W,L;S: AL,GU)
For all other groups:	(R,A,W,L,X,S: GU)

(Optional parameter.)

When a keyword parameter or keyword parameter group is omitted from the :NEWGROUP command, the corresponding default parameter takes effect.

EXAMPLE:

The Account Manager wants to create a new group named GNAME, having the password GPASS, and central processor and connect time limits of 2000 seconds and 12000 minutes respectively. He enters the following command:

```
:NEWGROUP GNAME; PASS=GPASS; CPU=2000; CONNECT=12000
```

Altering Groups

To change one or more attributes of a group, an Account Manager User can issue the :ALTGROUP command:

```
:ALTGROUP  groupname
     [;PASS = [password]]
     [;CAP = [capabilitylist]]
     [;FILES = [filespace]]
     [;CPU = [cpu]]
     [;CONNECT = [connect]]
     [;ACCESS = [fileaccess]]
```

groupname

The name of the group whose attributes are to be changed. (Required parameter.)

password
capabilitylist
filespace
cpu
connect
fileaccess

New values for the corresponding parameters described in the :NEWGROUP command. The *filespace* limit specified cannot be less than the number of sectors currently allocated for the group. (Optional parameters.)

When an entire keyword parameter group is omitted from an :ALTGROUP command, the corresponding value for the group remains unchanged. When a keyword is included but the corresponding parameter is omitted (as in ;PASS=), the default value is assigned.

When a parameter is modified through :ALTGROUP, it immediately takes effect in the directory but does not apply to current accesses under the group; thus, if an attribute is taken away from a group, users of the group currently running with that attribute retain it until again logging-on.

EXAMPLE:

To assign a new password of PASS2 to a group named GROUPXX, the Account Manager enters:

```
:ALTGROUP GROUPXX; PASS=PASS2
```

Listing Group Attributes

An Account Manager or System Manager User can print a list of attributes (previously specified in the :NEWGROUP or :ALTGROUP commands) for various groups, by issuing the :LISTGROUP command. (When this is done for all groups in the system, the attributes of the accounts to which the groups belong are also listed.)

:LISTGROUP [groupset] [, listfile]

groupset The set of groups to be listed, specified as follows:

Specification	Meaning
groupname.acctname	The group named, in the account named.
groupname	The group named, in the log-on account.
@.*acctname*	All groups in the account named.
@	All groups in the log-on account.
@.@	All groups in all accounts. (The accounts are also listed.)

If Account Manager Users who are not *also* System Manager Users specify an account, it must be their own.

If *groupset* is omitted, @ is assigned by default.

(Optional parameter.)

listfile The actual designator of the file on which the attribute listing is produced. The options assigned this file when it is opened (through the FOPEN intrinsic) are:

NEW, ASCII, VARIABLE, CCTL, OUTPUT (RETAIN DATA), EXCL

The file is closed (through the FCLOSE intrinsic) with SAVE disposition. If the *listfile* parameter is omitted, the job/session list device ($STDLIST) is assigned. (Optional parameter.)

EXAMPLE:

To list the attributes of all groups in the account named PROJECTS, the user enters:

```
:LISTGROUP @.PROJECTS
```

The group listing appears as an octal dump of the group entry or entries. Each entry is headed by the character *G* (for group), an equals sign, and the group name. The user information in the entry is decoded as shown below; both user and system information is decoded as shown in Appendix J.

Words	Content
0-3	Group name
5-8	Password
9-10	Permanent file space usage count (in sectors)
11-12	Permanent file space limit (in sectors)
13-14	Central-processor time usage count (in seconds).
15-16	Central-processor time limit (in seconds).
17-18	Connect-time count (in minutes).
19-20	Connect-time limit (in minutes).
21-22	Purge and group-security flags. (See Appendix K.)
23	Capability-class attributes (in the same bit-format returned by the WHO intrinsic, described in *Multiprogramming Executive Operating System*).

A translation of those bytes that contain alphanumeric ASCII characters appears to the right of each line in the octal dump.

EXAMPLE:

A user wants to list all groups in the account named DIAG. He enters the appropriate :LISTGROUP command, with the following result.

```
:LISTGROUP @.DIAG


G=  CURTIS
041525 051124 044523 020040 000345 042115 041123 044124 CURTIS....NONONO
020040 000000 000000 077777 177777 000000 000003 077777 ................
177777 000000 000025 077777 177777 002041 004102 000600 .............B..
000000                                                  ..


G=  DELANO
042105 046101 047117 020040 000337 042104 020040 020040 DELANO....DD....
020040 000000 000000 077777 177777 000000 000000 077777 ................
177777 000000 000000 077777 177777 002041 004102 000600 .............B..
000000                                                  ..


G=  LEMOS
046105 046517 051440 020040 000335 052117 050440 020040 LEMOS.....TOQ...
020040 000000 000000 077777 177777 000000 000000 077777 ................
177777 000000 000000 077777 177777 002041 004102 000600 .............B..
000000                                                  ..


G=  MACH
046501 041510 020040 020040 000631 044101 047101 020040 MACH......HANA..
020040 000000 000000 077777 177777 000000 000001 077777 ................
177777 000000 000001 077777 177777 002041 004102 000600 .............B..
000000                                                  ..


G=  PUB
050125 041040 020040 020040 000075 020040 020040 020040 PUB.............
020040 000000 000000 077777 177777 000000 000003 077777 ................
177777 000000 000000 077777 177777 020143 015006 000700 ...........C....
000000                                                  ..


G=  RADVANY
051101 042126 040516 054440 000341 042515 051131 020040 RADVANY...EMRY..
020040 000000 000000 077777 177777 000000 000000 077777 ................
177777 000000 000000 077777 177777 002041 004102 000600 .............B..
000000                                                  ..
```

Deleting Groups

To remove a group (and all files belonging to it) from the system, an Account Manager User enters:

```
:PURGEGROUP    groupname
```

groupname The name of the group to be removed. (Required parameter.)

When this command is entered in a session, MPE/3000 prints this message:

```
GROUP groupname TO BE PURGED?
```

The user verifies his intent to delete the group by responding YES; otherwise, he responds NO.

(No verification is made when the :PURGEGROUP command is entered in a job.)

If no files in the group are in use, *and* the group itself is not in use (as defined in the discussion of the :ALTACCT command), :PURGEGROUP removes the entire group. Otherwise, :PURGEGROUP removes every file not in use. In such a case, to completely purge the group, the user must re-issue the :PURGE command when neither the group nor its remaining files (if any) are in use.

The public group of the system account, PUB.SYS, cannot be purged.

EXAMPLE:

When an Account Manager User removes a group named EXGROUP from the system, this dialogue takes place:

```
:PURGEGROUP EXGROUP
GROUP EXGROUP TO BE PURGED? YES
```

Defining Users

An Account Manager User can define a new user through the :NEWUSER command. This user can log-on to any group for which he knows the password, or which is assigned as his home-group.

```
:NEWUSER    username
     [;PASS = [password]]
     [;CAP = [capabilitylist]]
     [;MAXPRI = [subqueuename]]
     [;LOCATTR = [localattribute]]
     [;HOME = [homegroupname]]
```

username — The name of the user, composed of up to eight alphanumeric characters, beginning with a letter. (Required parameter.)

password — The same meaning and default value as the corresponding parameter in the :NEWACCT command, but applicable to the user. (Optional parameter.)

capabilitylist — The same meaning as the *capabilitylist* parameter for the :NEWACCT command, but applicable only to this user. This parameter cannot specify capabilities not assigned to the Account (in :NEWACCT or :ALTACCT commands.) However, if the account is later altered, the user could be left with capabilities that exceed those of the account; but, because the user's capabilities are always verified to be a subset of the account's at log-on time, the user is never granted a capability not possessed by the account. If the *capabilitylist* parameter is omitted from the :NEWUSER command, the following capabilities are assigned by default: SF, IA, BA. (Optional parameter.)

subqueuename — The same meaning as the *subqueuename* parameter for the :NEWACCT command, but applicable only to this user. The priority specified in :NEWUSER cannot be greater than that specified in :NEWACCT or :ALTACCT. (But, if the user's account is later altered, he may be left with excessive limits.) The *subqueuename* defined for the user is checked against the subqueuename defined for his account at log-on time, and the lower priority of the two is used as the maximum priority restricting all processes of the job. Also, the priority requested by the user when he logs-on is checked against the *subqueuename* defined for him, and he is granted the lower of these two values. The default value is CS. (Optional parameter.)

localattribute — The local attribute defined at the installation; this must be a subset of the bit configuration for the *account's* local attribute. If omitted, 0 is assigned. (Optional parameter.)

homegroupname — The name of an existing group to be assigned as the homegroup for this user. If omitted, no home group is assigned, and so the user must always specify a group when he logs on. (Optional parameter.)

If the optional parameters or keyword parameter groups are omitted, the corresponding parameters specified in :NEWACCT are assigned by default.

EXAMPLE:

To define a new user as LHSMITH, and assign him a password of SMITTY and a homegroup of HOMGPX, the Account Manager User issues the following command. (The user will have the default capabilities and "CS" subqueue priority.)

```
    :NEWUSER LHSMITH; PASS=SMITTY; HOME=HOMGPX
```

Altering User Attributes

To alter the attributes currently defined for a user, an Account Manager User enters:

```
:ALTUSER    username
        [;PASS = [password] ]
        [;CAP = [capabilitylist] ]
        [;MAXPRI = [subqueuename] ]
        [;LOCATTR = [localattribute] ]
        [;HOME = [homegroupname] ]
```

username The name assigned to the user (in the :NEWUSER command). (Required parameter.)

password
capabilitylist
subqueuename
localattribute
homegroupname

The new values to be assigned the corresponding parameters originally specified in the :NEWUSER command. A System Manager User (SM) or Account Manager User (AM) cannot take away his own manager capability (Optional parameters.)

When an entire keyword parameter group is omitted from an :ALTUSER command, the corresponding value for the user remains unchanged. When a keyword is included but the corresponding parameter is omitted (as in ;CAP=), the default value is assigned.

When a parameter is modified through :ALTUSER, it immediately takes effect in the directory, but does not apply to users currently logged-on; it will take effect for the *next* log-on by those users.

EXAMPLE:

To change the capabilitylist *of a user (JONES) from* IA, BA, SF, PH, DS *to include the Multiple RIN capability, the Account Manager User enters:*

```
:ALTUSER JONES; CAP=IA,BA,SF,PH,DS,MR
```

Listing User Attributes

An Account Manager or System Manager User can list the attributes currently assigned to various users by issuing the :LISTUSER command. (When this is done for all users in the system, the attributes of the user's accounts are also listed.)

```
:LISTUSER [userset] [,listfile]
```

userset The set of users to be listed, specified as follows:

Specification	Meaning
username.acctname	The user named, in the account named.
username	The user named, in the log-on account.
@.acctname	All users in the account named.
@	All users in the log-on account.
@.@	All users in all accounts. (The accounts are also listed.)

If Account Manager Users who are not *also* System Manager Users specify an account, it must be their own.

If *userset* is omitted, @ is assigned by default.

(Optional parameter.)

listfile The actual designator of the file on which the attribute listing is produced. The options assigned this file when it is opened (through the FOPEN intrinsic) are:

NEW, ASCII, VARIABLE, CCTL, OUTPUT (RETAIN DATA), EXCL

The file is closed (through the FCLOSE intrinsic) with SAVE disposition. If the *listfile* parameter is omitted, the job/session list device ($STDLIST) is assigned.

EXAMPLE:

To list the attributes of the user named CLANCY in the account named MGT, the Account Manager User enters:

```
:LISTUSER CLANCY.MGT
```

The user listing appears as an octal dump of the user entry or entries. Each entry is headed by the character *U* (for user), an equals sign, and the user's name. The user information in the entry is decoded as shown below; both user and system information is decoded as shown in Appendix J.

Words	Content
0-3	User name
4-5	Capability (in the same bit-format returned by the WHO intrinsic, described in *Multiprogramming Executive Operating System, (03000-90005).*)
6-7	Local attributes

Words	Content
8-11	Password
12-15	Home group
16	Number of users logged-on
17	Maximum job priority (numerical)

A translation of those bytes that contain alphanumeric ASCII characters appears to the right of each line in the octal dump.

EXAMPLE:

A System/Account Manager User wants to list the attributes of all users in the account named LANG. He enters the appropriate :LISTUSER command, with the following result:

```
:LISTUSER @.LANG

U=  BAUSEK
Ø411Ø1 Ø52523 Ø42513 Ø2ØØ4Ø Ø31ØØ3 ØØØ7ØØ ØØØØØØ ØØØØØØ BAUSEK..2.......
Ø2ØØ4Ø Ø2ØØ4Ø Ø2ØØ4Ø Ø2ØØ4Ø Ø41517 Ø41117 Ø46Ø4Ø Ø2ØØ4Ø ........COBOL...
ØØØØØØ ØØØ226 ØØØØØØ                                    ......

U=  COUCH
Ø41517 Ø525Ø3 Ø44Ø4Ø Ø2ØØ4Ø ØØ1ØØ3 ØØØ6Ø1 ØØØØØØ ØØØØØØ COUCH...........
Ø2ØØ4Ø Ø2ØØ4Ø Ø2ØØ4Ø Ø2ØØ4Ø Ø411Ø1 Ø51511 Ø41517 Ø4652Ø ........BASICOMP
ØØØØØØ ØØØ226 ØØØØØØ                                    ......

U=  GREEN
Ø43522 Ø425Ø5 Ø47Ø4Ø Ø2ØØ4Ø ØØ1ØØ3 ØØØ6Ø1 ØØØØØØ ØØØØØØ GREEN...........
Ø2ØØ4Ø Ø2ØØ4Ø Ø2ØØ4Ø Ø2ØØ4Ø Ø411Ø1 Ø51511 Ø4144Ø Ø2ØØ4Ø ........BASIC...
ØØØØØØ ØØØ226 ØØØØØØ                                    ......

U=  MANAGER
Ø465Ø1 Ø471Ø1 Ø435Ø5 Ø51Ø4Ø Ø71ØØ3 ØØØ7Ø1 ØØØØØØ ØØØØØØ MANAGER.R.......
Ø46117 Ø41467 Ø3Ø462 Ø3ØØ4Ø Ø5Ø125 Ø41Ø4Ø Ø2ØØ4Ø Ø2ØØ4Ø LOC712Ø.PUB.....
ØØØØØØ ØØØØ36 ØØØØØØ                                    ......

U=  MARTIN
Ø465Ø1 Ø51124 Ø44516 Ø2ØØ4Ø ØØ1ØØ3 ØØØ6Ø1 ØØØØØØ ØØØØØØ MARTIN..........
Ø2ØØ4Ø Ø2ØØ4Ø Ø2ØØ4Ø Ø2ØØ4Ø Ø411Ø1 Ø51511 Ø4144Ø Ø2ØØ4Ø ........BASIC...
ØØØØØØ ØØØ226 ØØØØØØ                                    ......

U=  SHIPMAN
Ø5151Ø Ø4452Ø Ø465Ø1 Ø47Ø4Ø ØØ1ØØ3 ØØØ6Ø1 ØØØØØØ ØØØØØØ SHIPMAN.........
Ø2ØØ4Ø Ø2ØØ4Ø Ø2ØØ4Ø Ø2ØØ4Ø Ø411Ø1 Ø51511 Ø4144Ø Ø2ØØ4Ø ........BASIC...
ØØØØØØ ØØØ226 ØØØØØØ                                    ......
```

Deleting Users

To remove a user from the account, an Account Manager User issues the :PURGEUSER command.

```
:PURGEUSER    username
```

username The name of the user to be deleted. (Required parameter.)

When this command is entered in a session, MPE/3000 issues the following message:

USER username TO BE PURGED?

The Account Manager User must verify his intent by responding YES or NO.

(When the :PURGEUSER command is entered in a job, no verification is made.)

If a :PURGEUSER command references a user currently logged onto the system, that user is *not* purged and a message is output indicating this fact.

The original system manager user, MANAGER.SYS, cannot be purged.

Removal of a user may leave some files in an account without their creators, preventing anyone from renaming those files or altering their security. But the Account Manager User still can access those files, and can copy or purge them.

EXAMPLE:

When an Account Manager User removes the user SMITH from the system, this dialogue takes place:

```
:PURGEUSER SMITH
USER SMITH TO BE PURGED? YES
```

COMMAND SUMMARY

The MPE/3000 System/Account Manager Commands are summarized below according to the user attribute (System Manager or Account Manager) required to enter them.

System Manager (SM) Capability

:ALTACCT
:LISTACCT (All accounts)
:LISTGROUP (All accounts)
:LISTUSER (All accounts)
:NEWACCT
:PURGEACCT

Account Manager (AM) Capability

:ALTGROUP
:ALTUSER
:LISTACCT (User's own account)
:LISTGROUP (User's own account)
:LISTUSER (User's own account)
:NEWGROUP
:NEWUSER
:PURGEGROUP
:PURGEUSER

SECTION VII
Using the Accounting System

The MPE/3000 Accounting System keeps track of various system resources used by each account and group; these resources are total permanent file space and central processor time (accumulated by jobs and sessions), and terminal connect time (accumulated by sessions). Furthermore, limits can be set for the maximum use of these resources at the account level (by System Manager Users issuing the :NEWACCT or :ALTACCT commands) and at the group level (by Account Manager Users issuing the :NEWGROUP and :ALTGROUP commands).

When a job/session is in progress, MPE/3000 maintains counts of the time-resources used by that job/session. As the job/session is logged off, its total time-resource use counts are used to update the time resource-use counters for its log-on account and group. When another job/session attempts to log-on to the same account (and perhaps group), and the central-processor time limit or session connect-time limit has been exceeded by the previous job/session at the account (or group) level, access is refused.

When a request is made to save a file, or add an extent to an existing file, and this action would result in exceeding the permanent file space limit at either the account or group level, the request is denied. (File space is always charged to the group containing the file, rather than the group where the user who created the file was logged-on.)

DISPLAYING ACCOUNTING INFORMATION

The accounting information for each group and account can be extracted and displayed, showing counts and limits for permanent file space (in disc sectors), central processor time (in seconds), and session connect-time (in minutes). System Manager Users can extract this information for any individual accounts and groups, or for all groups under all accounts; Account Manager Users can display this information for any or all groups in their log-on account. (For any particular resource, the sum of all group counts within any account always equals the total count accumulated by the account.) The accounting information is requested with the :REPORT command, described below. (A more restrictive version of this command is available to the standard user for displaying the total accounting information for his log-on account and group. This is discussed in *HP 3000 Multiprogramming Executive Operating System (03000-90005).*) The accounting information can be used for billing or for simply obtaining an overview of system usage on an account/group basis.

NOTE: *In the counts of central processor and connect time used, the values shown reflect time accumulated* prior to *the current log-on; the amounts accumulated by the current job/sessions are not included.*

The :REPORT command is entered in the following format.

:REPORT [groupset] [,listfile]

groupset — Entries that specify the accounts and groups for which the resource-use report is generated, as described below. If no entry is included, the report covers all groups in the log-on account. (Optional parameter.)

listfile — An actual file designator from the output set that denotes the file to which the report is written. If omitted, the job/session listing device ($STDLIST) is used. For *listfile*, the options supplied (by the FOPEN intrinsic) when the file is opened are:

NEW, ASCII, VARIABLE, CCTL, OUTPUT (RETAIN DATA)

The *listfile* is closed (through the FCLOSE intrinsic) with the NO CHANGE disposition.
(Optional parameter.)

The *groupset* entries used to specify the accounts and groups covered by the report are denoted below. Certain entries can be entered only by the System Manager User, while others are permitted to both System and Account Manager Users as noted below:

Parameter Entry	Group Covered by Report	Account Covered by Report	Who may enter this Combination
groupname	The group named.	The log-on account.	System or Account Manager.
@	All groups.	The log-on account.	System or Account Manager.
groupname.acctname	The group named.	The account named.	System Manager.
@.acctname	All groups.	The account named.	System Manager.
@.@	All groups.	All accounts.	System Manager.
(omitted)	All groups.	The log-on account.	Any Manager User.

The type of output written to *listfile* depends on the type of file (ASCII or binary) specified or implied. If *listfile* is an ASCII file, a standard ASCII listing is produced; on this listing, an unlimited quantity is denoted by a double asterisk (**). If *listfile* is a binary file, (typically used to help in automatic processing of the report data), a 17-word record is written for each account/group. This record is decoded as follows:

Word	Contents
0	Type of entry, where: 1 = A group entry. 2 = An account entry.

Word	Contents
1-4	Account or group name, left-justified and padded with blanks.
5-6	Permanent file space count (in sectors).
7-8	Permanent file space limit (in sectors).
9-10	Central processor time count (in seconds).
11-12	Central processor time limit (in seconds).
13-14	Connect time count (in minutes).
15-16	Connect time limit (in minutes).

On a binary *listfile*, counts and limits are double-word integers; an unlimited quantity is denoted as %17777777777.

If the type of file for *listfile* is not specified, ASCII is assigned by default.

On both ASCII and binary *listfiles*, the entry for each account is immediately followed by the entries for all of its groups.

To create permanent *listfiles*, the :FILE command should be used as shown in the examples below.

EXAMPLES:

To create a permanent ASCII file (REPFILE) showing resource-usage counts and limits for all groups in the log-on account, the Account Manager User enters:

```
:FILE REPFILE; SAVE; REC=66
:REPORT @, *REPFILE
```

To create a permanent binary file (REPFILE) showing resource usage counts and limits for all accounts/groups in the system, the System Manager User enters:

```
:FILE REPFILE; SAVE; NOCCTL; REC=17,,,BINARY
:REPORT @.@, *REPFILE
```

An example of a report covering all groups in the account named DIAG, generated on an ASCII file, appears below:

```
:REPORT @.DIAG
ACCOUNT          FILESPACE-SECTORS       CPU-SECONDS        CONNECT-MINUTES
    /GROUP         COUNT    LIMIT      COUNT     LIMIT      COUNT     LIMIT

DIAG                7734     **         8288      **          184       **
    /CURTIS            Ø     **            3      **           21       **
    /DELANO            Ø     **            Ø      **            Ø       **
    /GIBBONS           Ø     **            Ø      **            Ø       **
    /GRAZIANO          Ø     **          216      **           68       **
    /HUNT              Ø     **           62      **           24       **
    /JOHNSTON       34ØØ     **         1974      **            1       **
    /LEMOS             Ø     **            Ø      **            Ø       **
    /MACH              Ø     **            1      **            1       **
    /PUB               Ø     **            3      **            Ø       **
    /RADVANY           Ø     **            Ø      **            Ø       **
    /WOLFF          4334     **         6Ø29      **           69       **
```

RE-SETTING RESOURCE-USE COUNTS

The System Manager (but not the Account Manager User) can reset to zero the running counts of central processor time or connect-time accumulated by an account, and all groups within that account. Typically, he may do this after a resource-use report has been obtained and billing has been accomplished.

```
:RESETACCT  [ @        ]  [ ,CPU     ]
            [ acctname ]  [ ,CONNECT ]
```

@ = A specification that counters for all accounts/groups are to be reset. (Optional parameter.)

acctname = The name of a particular account whose account and group usage counters are to be reset. (Optional parameter.)

CPU = A specification that *only* the central processor usage counter is to be reset. (Optional parameter.)

CONNECT = A specification that *only* the connect-time usage counter is to be reset. (Optional parameter.)

If neither @ nor *acctname* is specified, the designated counters for all accounts (and their groups) are reset (@). If neither *CPU* nor *CONNECT* is specified, both the central processor and connect-time counters for the designated account (and its groups) are reset. If all parameters are omitted, all counters for all accounts and their groups are reset.

EXAMPLE:

A System Manager User can reset the central processor unit resource use counter for all accounts in the system by entering:

```
:RESETACCT @, CPU
```

SECTION VIII
Operational Management

In addition to all commands available to the standard user, the user with System Supervisor Capability can enter various commands which enable him to have supervisory operational control of the system. These commands, discussed in this section, can be entered in either session or batch mode. The session mode allows the user the option of requesting functions from a remote terminal.

MANAGING THE MASTER QUEUE

All processes competing for the central processor access it in an orderly manner, under the direction of MPE/3000. The system places all processes in the master scheduling queue in order of their priority. When a process in execution is completed, terminated, or suspended, the MPE/3000 Dispatcher searches the master queue for the next process of highest priority, and transfers control to that process.

The master queue (Figure 8-1) is divided into logical areas, each corresponding to a particular type of dispatching and priority class for the processes within it. A logical area can be a linear subqueue, a circular subqueue, or a portion of the main master queue. In a linear subqueue, the process with highest priority accesses the central processor first and maintains this access until the process either is completed, terminated, or suspended to await the availability of a required resource. In a circular subqueue, all processes have equal priority and each accesses the central processor for an interval (time quantum) of maximum duration (or until completed, terminated, or suspended). At the end of this duration, control is transferred to the next process in the queue, continuing in a round-robin fashion. This time-slicing is controlled by the system timer. Processes that are not scheduled in a subqueue are scheduled in the master queue.

Each subqueue in the master queue is defined by a priority number. While the standard user is aware of the priority class associated with a subqueue, only a user with System Supervisor or Privileged Mode Capability can deal with priority numbers. The standard subqueues (priority classes) are as follows:

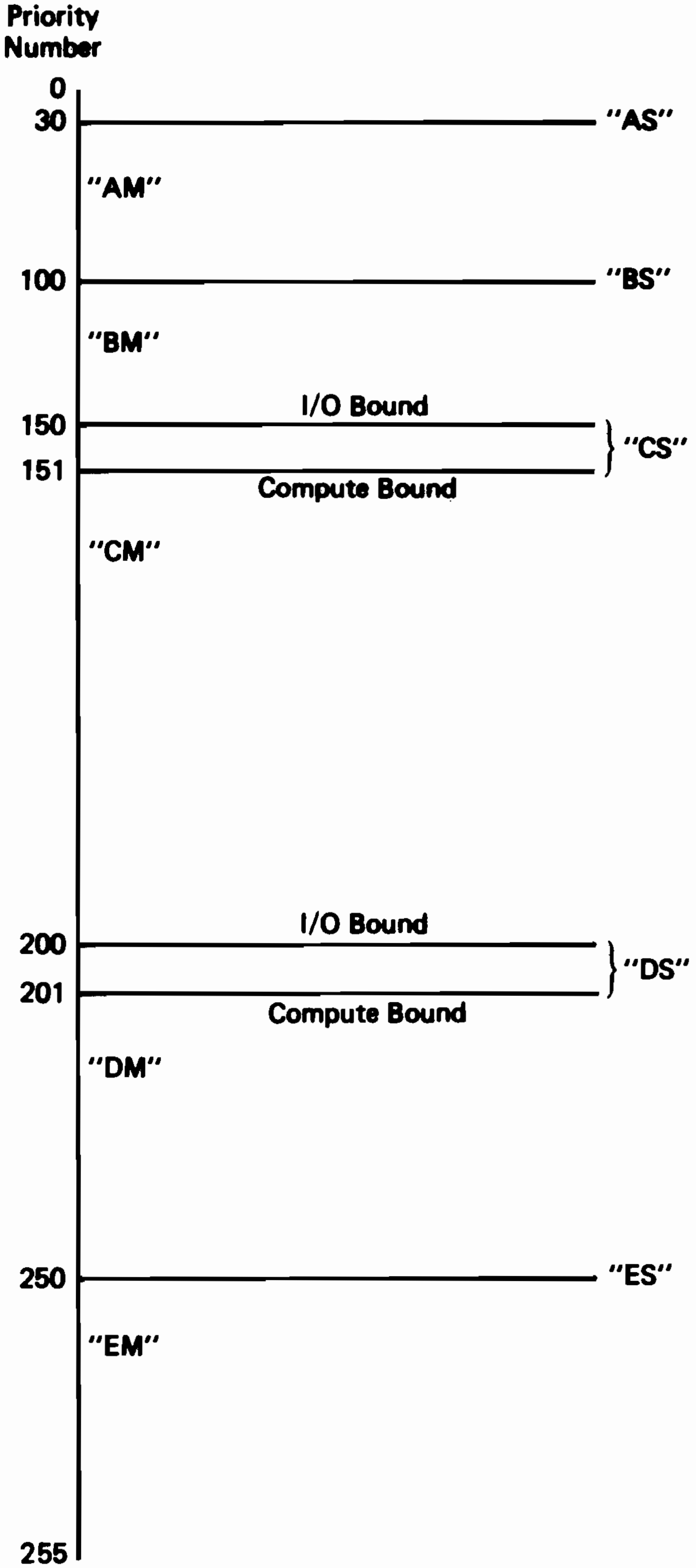


Figure 8-1. MPE/3000 Master Queue Structure

AS

is a linear subqueue containing processes of very high priority. (In a subsequent MPE/3000 release, it will be accessible only to users with a special *MPE/3000 Optional Capability* to be added to the system then.) Its priority number is 30.

BS

is a linear subqueue containing processes of high priority. (In a subsequent MPE/3000 release, it will be accessible only to users having a special *MPE/3000 Optional Capability* to be added to the system.) Its priority number is 100.

CS

is composed of two circular sub-subqueues, each having a time-quantum determined at system generation time. This subqueue is used mainly for interactive processes and for multiprogramming batch jobs. The time quantum specifies the maximum execution time allowed any process in the sub-subqueue before going on to the next process. The priority numbers of the sub-subqueues are 150 and 151; the higher-priority sub-subqueue is used for input/output bound processes and the lower-priority sub-subqueue is used for compute-bound processes.

DS

is also composed of two circular sub-subqueues, each having a time-quantum determined at configuration time. This subqueue is available for general use at a lower priority than the CS subqueue. The priority numbers of the sub-subqueues are 200 and 201; as in the CS subqueue, one sub-subqueue is used for input/output bound processes and the other is used for compute-bound processes.

ES

is a linear subqueue, primarily containing idle processes with low priority. Its priority number is 250.

AM,BM,CM,DM,EM

are discontiguous portions of the main master queue. Processes can be scheduled in these areas only if they are system processes or are initiated by a user with Privileged Mode Capability. Such processes are dispatched linearly.

The priority class of a process can be specified by the normal user with standard or optional MPE/3000 capabilities. In the two-character string that comprises a priority-class reference, the first character refers to the location of a subqueue within the master queue (in alphabetical order) and the second character specifies whether the logical area is the subqueue itself (S) or the portion of the master queue (M) that immediately follows the subqueue.

Priority numbers range from 1 to 255 inclusively, with 1 denoting the highest priority. Any two processes have the same priority number are scheduled in the same subqueue. Only the System Supervisor User can specify the relative ranks of processes having identical numbers within a linear subqueue; only the Privileged Mode User can request a portion of the master queue or *specify an absolute* priority number for a process.

With each circular subqueue of priority P, is associated another subqueue of priority P-1 called the input/output subqueue. (As noted above, these are the *sub-subqueues* within the CS and DS classifications.) If any process is detected as being input/output bound, the system switches it from the normal (compute-bound) subqueue to the input/output subqueue. (An input/output bound process is defined as one which, for a specified number of consecutive times, has been suspended for input/output instead of being timed out at the end of its time-quantum.) This gives the process a higher priority until it is no longer input/output bound. A process is considered to be compute-bound when, for a specified number of consecutive times, it reaches the end of its time-quantum. Users cannot directly access input/output bound subqueues — processes are always moved to and from these subqueues by the system to improve response time for input/output interactive processes. The time quantum on each input/output bound subqueue is also dynamically-adjusted to regulate the load on each queue.

A System Supervisor User can control the master queue in the following ways:

- Modify the time-quantum for any circular subqueue, on-line, as discussed below.

- Display the master queue or subqueues, as described in the following pages.

Changing a Circular Subqueue Quantum

The System Supervisor User can change the value of the quantum of a circular subqueue by issuing the :QUANTUM command. This command is used primarily in on-line tuning of the system response time to best accommodate the current load.

> *NOTE: Misuse of this command can significantly degrade system operating efficiency.*

The command format is

```
:QUANTUM subqueuename,time
```

subqueuename The first letter in the name of the subqueue. (Required parameter.)

time The new time-quantum, in milliseconds, for this subqueue. This value must lie between 0 and 65,535 milliseconds. (Required parameter.)

EXAMPLE:

To change the quantum of the CS subqueue to 100 milliseconds, the user enters:

```
:QUANTUM C, 100
```

Displaying Subqueue Information

A System Supervisor User can display information about the scheduling of processes and the contents of various scheduling subqueues or the master queue, by entering the :SHOWQ command.

:SHOWQ [subqueuename]

subqueuename The first letter in the name of the subqueue to be displayed. If this parameter is included, the following information is displayed:

- The name of the subqueue.
- An optional dollar sign ($) output to indicate the higher-priority input/output-bound subqueue if *subqueuename* referenced a double subqueue.
- A list of the processes currently residing in the subqueue, by process identification number (PIN). The prefix *M* indicates a job or session main process.

If the *subqueuename* parameter is omitted, the following information is displayed for the *master queue:*

- The priority number of every process (P) in the master queue.
- The priority number of every standard subqueue (*x*S), where *x* is the first letter of the subqueue name, plus:

    An indication of whether the subqueue is circular (*C*) or linear (*L*).

    The time-quantum, in milliseconds, if defined for a circular subqueue.

    An indication (EMPTY) if no processes reside in the subqueue.

(Optional parameter.)

EXAMPLES:

To display the contents of the CS subqueue, the user enters:

```
:SHOWQ C
```

As a result, the following information is printed. (The two-digit numbers are PIN's and M indicates a job or session main process.)

```
SQ -C-  $   M32 .
SQ -C-      M25 .M27 .M26 .M28 .M33 . 31 . 29 . 3Ø .
```

To display the overall structure of the master queue's subqueues, the user enters:

```
:SHOWQ
```

The following type of information is output, where asterisks indicate standard subqueues:

```
PRIORITY     NAME     C/L      TQ(MS)

   5         P
   6         P
  19         P
  2Ø         P
  21         P
  22         P
  24         P
  25         P
  26         P
  27         P
  28         P
  29         P
  3Ø         AS *      L                    EMPTY
  5Ø         P
  6Ø         P
  1ØØ        P
  1ØØ        BS *      L                    EMPTY
  12Ø        P
  121        P
  122        P
  15Ø        CS *      C       1ØØØ
  151        CS *      C       1ØØØ
  2ØØ        DS *      C       1ØØØ         EMPTY
  2Ø1        DS *      C       1ØØØ         EMPTY
  25Ø        ES *      L                    EMPTY
```

CREATING PROCESSES

Processes are created by calling the CREATE intrinsic, available only to users with the *MPE/ 3000 Process-Handling Optional Capability*. A user with System Supervisor Capability, in addition, can specify the relative ranks of processes having identical priority numbers within a linear subqueue; this is done through the rank parameter of the CREATE intrinsic, available only to users with this capability.

The CREATE intrinsic loads the program to be run by the new process into virtual memory, creates the new process as the son of the calling process, initializes its data stack, schedules the process, and returns its process identification number (PIN) to the calling process.

The CREATE intrinsic is written in the following format:

PROCEDURE *CREATE (progname,* entryname, *pin,* param, flags, stacksize, dlsize, maxdata, priorityclass, rank;)

VALUE param, stacksize, dlsize, priorityclass, maxdata, flags, rank;

LOGICAL priorityclass, flags;

INTEGER stacksize, dlsize, maxdata, *pin,*param, rank;

BYTE ARRAY *progname,* entryname;

OPTION VARIABLE, EXTERNAL;

The parameters for this intrinsic call are as follows. All parameters except *progname* and *pin* are optional.

progname A byte array, containing a string terminated by a blank, specifying the name and optionally, the account and group of the file containing the program to be run.

entryname A byte array, containing a string terminated by a blank, specifying the entry-point (label) in the program where execution is to begin when the process is activated. The *primary* entry-point in the program can be specified in this array by a blank character alone. If entryname is omitted, the primary entry-point is specified by default.

pin A word in which the PIN of the new process is returned to the calling process. This PIN is used in other intrinsics to reference the new process. The PIN can range from 1 to 255. (If an error is detected, a PIN of zero is returned.)

param

A word used to transfer control information to the new process. Any instruction (in the outer block of code) in the new process can access this information in Location (Q-4). If param is omitted, this word is filled with zeros.

flags

A word whose bits, if on, specify loading options:

Bit (15:1) = *ACTIVE Bit.* If this bit is *on*, MPE/3000 re-activates the calling process (father) when the new process terminates. If this bit is *off*, the calling process is not activated at that time. The default setting is *off.*

Bit (14:1) = *LOADMAP Bit.* If this bit is *on*, a listing of the allocated (loaded) program is produced on the job/session listing device. This map shows CST entries used by the new process. If this bit is *off*, no map is produced. The default setting is *off.*

Bit (13:1) = (Not used.)

Bit (12:1) = *NOPRIV Bit.* If this bit is *on*, the program is loaded in *non-privileged mode.* If this bit is *off*, the program is loaded in the mode specified when the program file was prepared. The default setting is *off.*

Bits (10:2) = *LIBSEARCH Bits.* These bits denote the order in which libraries are to be searched for the program:

00 = System Library

01 = Account Public Library, followed by System Library.

10 = Group Library, followed by Account Public Library, followed by System Library.

The default setting for these bits is *00.*

Bits (0:10) = (Not used by MPE/3000)

If *flags* is omitted, all default values noted are taken.

stacksize

An integer (Z-Q) denoting the number of words assigned to the local stack area (the area bounded by the initial Q and Z registers). The default value is that specified during system generation.

dlsize — An integer (DB-DL) denoting the number of words in the user-managed stack area (bounded by the DL and DB registers). The default value is that specified in the program file.

maxdata — The maximum size allowed for the process' stack (Z-DL) area in words. When specified, this value overrides the one established at program-preparation time. The default value is specified at system generation-time.

priorityclass — A string of two ASCII characters describing the priority class (subqueue) in which the new process is scheduled. (This may be "AS", "BS", "CS", "DS", or "ES".) Users with Privileged Mode Capability can specify priority numbers, or "AM", "BM", "CM", "DM", or "EM", as directed in *HP 3000 Multiprogramming Executive Operating System (03000-90005).* The default value is the priority of the calling process.

rank — The relative rank of the process within a linear subqueue. (Processes in a linear subqueue are linked together, defining an implicit order (1, 2, 3, and so forth). The *rank* parameter defines the *relative* location of a process in the chain; it is not kept as a fixed attribute of the process. If omitted, the process is scheduled as the last of the chain.

When the CREATE intrinsic is called, the DB register must be pointing to the user's stack.

The CREATE intrinsic returns one of the following condition codes to the calling process:

CCE — Request was granted; the new process is created.

CCG — Request was granted; the *maxdata* and/or *dlsize* parameters given were illegal, but other values were used. Specifically,

1. If the *maxdata* specified exceeds that maximum Z-DL allowed by the configuration, then that configuration maximum value is assigned.
2. If (*dlsize* + 100) modulo 128 is *not* zero, then *dlsize* is rounded upward so that (*dlsize* + 100) modulo 128 = 0.

CCL — The request was not granted because the *progname* or *entryname* specified does not exist.

The creating process is aborted if:

- Request was rejected because of illegal parameters; a PIN of zero is returned. Specifically, this occurs,

  If *progname* is illegal.

  If *entryname* is illegal.

If *stacksize* is less than 512 (decimal) and is not -1. (Note that -1 specifies the default value.)

If *dlsize* is less than 0 and is not -1.

If *maxdata* is less than or equal to 0, and is not -1.

If (*dlsize* + *globsize* + *stacksize* + 128) exceeds *maxdata.* Note that *dlsize* may have been modified to satisfy Condition 2, under CCG, described above. (The *globsize* value is the sum of the primary DB plus the secondary DB values — the total DB given at program preparation time by the program map (PMAP).)

If (*dlsize* + *globsize* + *stacksize* + 128) exceeds the maximum *stacksize* defined during system configuration. Note that *dlsize* may have been modified to satisfy Condition 2, under CCG, described above.

If (*maxdata* + 90) exceeds 32,768, where *maxdata* is either the value passed as a parameter or a value re-computed by the Loader under Condition 1 of CCG (above).

- The user does not have the *Process-Handling Optional Capability.*
- An illegal value (a non-existent subqueue) was specified for the *priorityclass* parameter.
- A required parameter (*progname* or *pin*) is omitted.
- A reference parameter was not within the required range.

EXAMPLE:

Suppose the user wants to create a process that, when activated, runs a program in the file PROGA, beginning at the entry-point START1. He also wants the libraries searched in the following order: account public library, followed by system library. The process' PIN is to be returned to the word PINRET. The process is to be assigned the priority class ES, and rank 2. The user enters the following intrinsic call. (The byte array PROG contains "PROGA", the byte array ENTER contains "START1," the tenth bit of the word FLAGGER is on, and the word RNK contains "2".)

```
CREATE (PROG,ENTER,PINRET,,FLAGGER,,,,"ES",RNK);
```

The PIN assigned to the process is returned to the word PINRET.

> *NOTE: Users with privileged-mode capability should note the following: If the master queue is specified in the* priorityclass *parameter, and if the* rank *specified leads to a conflict in priority with another process, the priority of the new process is adjusted so that this process is scheduled in the first location available, moving toward the ES subqueue.*

RESCHEDULING PROCESSES

When a process is created, it is scheduled on the basis of a priority-class assigned by its father. After this point, its priority-class can be changed at any time through the GETPRIORITY intrinsic. (This intrinsic is restricted to users with the *MPE/3000 Process-Handling Optional Capability*; users also having the System Supervisor Capability can use the *rank* parameter of this intrinsic to specify the relative rank of a process within a linear subqueue.) The intrinsic can be called from a process to change the priority of that process or of one of its sons—but not the priority of that process' father.

The format of the GETPRIORITY intrinsic call is

```
PROCEDURE   GETPRIORITY (pin, priorityclass, rank);

VALUE       pin, priorityclass, rank;

LOGICAL     priorityclass;

INTEGER     pin; rank;

OPTION VARIABLE, EXTERNAL;
```

The parameters for the GETPRIORITY intrinsic are

pin — An integer denoting the process whose priority-class is to be changed. If this is a son process, the integer is the process' PIN. If this is the calling process, the integer is zero.

priorityclass — A string of two ASCII characters describing the priority-class (subqueue) in which the process is rescheduled. (This may be "AS", "BS", "CS", "DS", or "ES".) Users with Privileged Mode Capability can specify priority numbers or "AM", "BM", "CM", "DM", or "EM", as directed in *HP 3000 Multiprogramming Executive Operating System (03000-90005).*

rank — The relative rank of the process within a linear subqueue, as defined in the discussion of CREATE. If omitted, the process is scheduled in the last position in the subqueue.

The GETPRIORITY intrinsic returns one of the following condition codes:

CCE	Request granted.
CCG	The process is not alive.
CCL	Request denied because an illegal PIN was specified.

The process is aborted if the user does not have the System Supervisor or Process-Handling Capability, or specifies a non-existent subqueue.

EXAMPLE:

To reschedule a process with the priority class "ES," and a rank of 3, a user issues the following call from within that process.

```
GETPRIORITY (Ø,"ES",3);
```

NOTE: Users with privileged Mode Capability should note the following: If the master queue is specified in the priorityclass *parameter, and if the* rank *specified leads to a conflict in priority with another process, the priority of the new process is adjusted so that it is scheduled in the first location available, moving toward the ES subqueue.*

PERMANENTLY ALLOCATING PROGRAMS AND PROCEDURES

A user with System Supervisor Capability can enhance system efficiency by storing program files (sets of one or more segments) or procedures (individual segments) permanently in virtual memory. This is, in effect, a permanent allocation of the programs or procedures so that they always reside in virtual memory regardless of whether they are being referenced. Since allocation of a code segment always results in the allocation of external segments referenced by it, when a program or procedure is permanently allocated, all of its external routines are also permanently allocated.

By permanently allocating large, frequently-used routines, a System Supervisor User can greatly reduce the system overhead that would otherwise be used in continually inserting and withdrawing them from virtual memory. Additionally, permanent allocation reduces the number of segment linkage operations required.

Allocation

To permanently allocate a program file or procedure, the user enters the :ALLOCATE command:

:ALLOCATE [[code],] name

code A directive specifying whether a program file or a code segment containing a procedure is to be allocated.

If PROGRAM is entered for this parameter, a program file (indicated by the *name* parameter) is allocated. (This program is loaded in the normal mode, with LIBSEARCH = S.)

If PROCEDURE is entered for this parameter, the code segment containing the specified procedure (indicated by the *name* parameter) that resides in the system segmented library (SL.PUB.SYS) is allocated.

If the *code* parameter is omitted, the default assigned is PROGRAM. (Optional Parameter.)

name The name of the program file or procedure to be allocated. If PROCEDURE is specified for the *code* parameter, the *name* parameter must be a procedure name. (Required parameter.)

The user issuing the :ALLOCATE command must pass all security provisions for any file referenced by the *name* parameter of this command. In other words, he must have execute-access to this file and he must know the lockword.

EXAMPLE:

To permanently allocate a program residing on a program file named RUNNER, in the PUB group of the SYS account, the user enters:

```
:ALLOCATE PROGRAM, RUNNER.PUB.SYS
```

To permanently allocate a procedure identified as PROC1, the user enters:

```
:ALLOCATE PROCEDURE, PROC1
```

De-Allocation

To remove from virtual memory a program file or procedure segment that has been permanently allocated, a System Supervisor User issues the :DEALLOCATE command. The program file or procedure segment is de-allocated immediately if it is not being used by a process; otherwise, it is de-allocated as soon as it is no longer in use. The :DEALLOCATE command format is

:DEALLOCATE[[code],] name

code A directive specifying whether a program file or procedure code segment is to be de-allocated.

If PROGRAM is entered for this parameter, a program file (indicated by the *name* parameter) is de-allocated.

If PROCEDURE is entered for this parameter, a code segment (indicated by the *name* parameter) is de-allocated.

If *code* is omitted, PROGRAM is assumed. (Optional parameter.)

name The name of the program file or procedure to be de-allocated. (Required parameter.)

SECTION IX
Using the Logging System

In addition to the Accounting System, MPE/3000 provides another facility that can be used for monitoring the use of certain resources by accounts, groups, and users — the Logging System. While the Accounting System is managed by System and Account Manager Users, the Logging System is managed by System Supervisor Users. Like the Accounting System, the Logging System can be used for billing or for simply obtaining an overview of system usage. Unlike the Accounting System, however, the Logging System describes system usage on a job/session basis; in fact, it provides the only means for doing so, and for monitoring certain dynamic resources. It creates a running log of actual events, correlated with the job/session that caused each event. The events monitored are:

- Job/Session Initiation
- Job/Session Termination
- Process Termination
- File Closing
- System Shut-Down

The running log is produced on one or more disc files known as *Log Files.* The events monitored are recorded on *Log-Records* within such files, (one logical record for each event). The System Supervisor User can supply configuration information for Log Files, create, close, and purge such files, and display the amount of space used on the Log File currently being written. However, the contents of Log Files — the individual Log Records — are not displayed nor otherwise used by MPE/3000. Instead, various analysis routines must be written by the user and supplied for this purpose.

LOG FILES

The Logging System is activated during system configuration, through the Configurator/User Dialogue (Steps 6 through 6.4) discussed in Section IV. These steps allow the user to:

- Enable or disable logging. (This option is also available during cold-loading, in Step 39 of the User/Initiator Dialogue discussed in Section V.)
- Select the events to be logged.

- Determine the physical Log-Record size (in sectors).
- Determine the Log-File size (in physical records).

When logging is enabled for the first time, the first Log File is created and opened on disc, and records are written to it by the Logging System as the events to be logged occur. When this Log File is full, it is closed by the system and a new Log File is created and opened on disc. This new file becomes the *current* Log File, and the previous Log File is de-activated. Also, when a Log File is closed by the System Supervisor User, or when the system is shut-down and cold-loaded again, the current Log File is de-activated and a new Log File is created and becomes the current Log File. Thus, several Log Files can exist in the system at any time; one is the *current* Log File, always an open file on disc, and the others are de-activated Log Files previously closed. The de-activated Log Files may exist on disc or may be copied to tape and optionally purged from disc, at the user's option.

A Log File is always designated by the name *LOGxxxx.PUB.SYS*, where *xxxx* is the Log File Number, ranging from 0000 to 9999; 0000 designates the first Log File. With each new Log File, *xxxx* is increased by 1.

Each time a new Log File is created, the following error message is output, showing the number (*x*) of the new (current) Log File:

LOG FILE NUMBER x ON

General Format

All Log Files are initially created as files containing fixed-length records, but are later transformed into files formatted for variable-length records. They should always be treated as files containing variable-length records, accessed sequentially.

For a Log File, the end-of-file pointer can point at the last record (block) written to the file (if the file is closed normally) or at any point beyond the last record written (if the system has not been shut down); in the latter case, all space following the last record is padded with zeros.

The general format of a Log File is shown in Figure 9-1.

Configurable Characteristics

As noted above, the user can specify, during system configuration, the physical Log Record (block) size and the total Log File size. These characteristics are specified in Steps 6.3 and 6.4, respectively, of the Configurator/User Dialogue.

The *block size* is designated in disc sectors, and cannot exceed 1024 words. Since the Logging System uses an internal double-buffering system, block size is equivalent to the size of the Log File buffer.

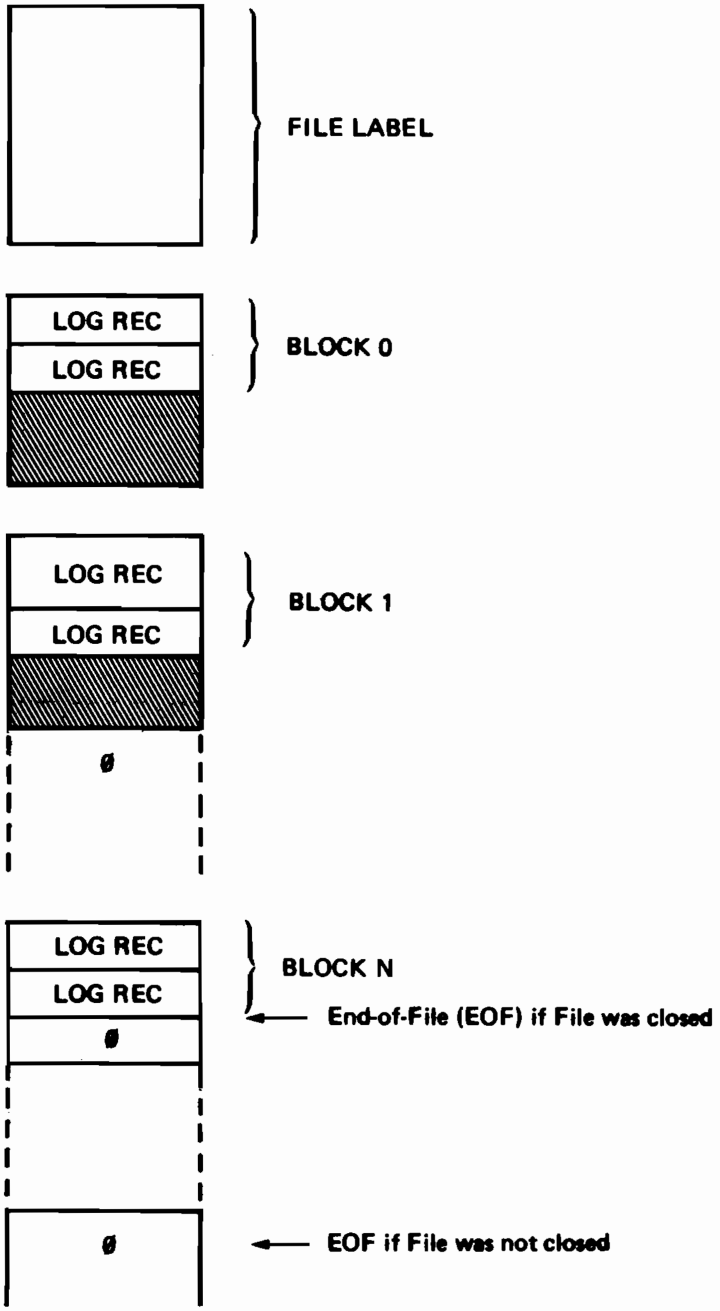


Figure 9-1. Log File Format

The *file size* is specified in number of blocks (physical records). The Log File is always created with 16 disc extents; one extent is allocated initially, and additional extents are allocated one at a time, as needed. The number of sectors in each extent (except possibly the last) is determined by the following formula:

$$\text{Sectors} = \left[\left(\frac{\text{Log File Size}}{16}\right) \times \left(\text{Log Record Size}\right)\right]$$

File Security

Because Log Files are files belonging to and created by MPE/3000 (Logging System), their creator, by implication, is the original System Manager User (MANAGER.SYS). A Log File is created with the MPE/3000 default security provisions; since it is in the PUB.SYS group/account, this implies that it can be read by any user but can be changed only by Account Librarian (AL) users for SYS, and by group users (GU) for PUB.SYS. Once the Log File is no longer current, the System Manager User can re-name the file (perhaps adding a *lockword*), and can re-define its security provisions. *Thus, the System Manager User controls access to the current Log File, but the System Supervisor User manages the Logging Facility by specifying Log File size, Log Record size, and events logged, and by enabling/disabling logging.*

Displaying Log File Status

At any time, the System Supervisor User can display the current Log File's number and the amount (percentage) of available file space currently used. He does this by entering

```
:SHOWLOG
```

The information displayed appears in this format:

```
LOGFILE LOGxxxx IS yy% FULL
```

xxxx	Log File Number
yy	Percentage of file space used.

If the Logging System is disabled, MPE/3000 outputs the message

```
NO LOGGING
```

If logging is enabled but currently suspended as a result of a managerial error (described later in this section), both messages appear.

EXAMPLE

To display the number and percent of file space used for the current Log File, the user enters:

```
:SHOWLOG
```

The following display appears:

```
LOG FILE LOG0655 IS 2% FULL
```

Creating a New Log File

The System Supervisor User can close the current Log File and create a new Log File by entering:

```
:SWITCHLOG
```

One or both of the messages described under *:SHOWLOG* appear, the current Log File is closed, a new Log File is created and opened, and the new Log File becomes the current Log File.

If Logging is disabled when *:SWITCHLOG* is entered, no action is taken.

File Error Handling

Two types of errors can prevent the Logging System from maintaining the Log File:

- *Catastrophic Errors.* These are physical input/output errors or unit failures. These errors are *not* recoverable; when they are detected, logging is disabled until the next cold-load.

- *Managerial Errors.* These are errors involving the creation and management of the Log File through the MPE/3000 File Management System. These errors are sometimes recoverable if the System Manager or System Supervisor User takes appropriate action. When such errors are detected, logging is temporarily suspended until recovery action appropriate to the error is completed and the System Supervisor User enters the command.

  ```
  :RESUMELOG
  ```

  When logging resumes, a special Log Record is emitted that denotes the number of log events (and corresponding records) missed: total records missed, total job/session initiation records missed, and total job/session termination records missed.

Log file errors are discussed further at the end of this section.

LOG RECORDS

Log Records are written to the Log File by MPE/3000. (This is done by calls to a special log intrinsic from the intrinsic or process that requires the recording of a particular event.) (The Log Records can be subsequently accessed, manipulated, and displayed through user-supplied analysis routines.)

Seven types of Log Records can be emitted:

Type No.	Type
0	Log Failure Record
1	Head Record
2	Job Initiation Record
3	Job Termination Record
4	Process Termination Record
5	File Close Record
6	Shut-Down Record

When Logging is enabled, Log Failure and Head Records are always output; the remaining types of records may or may not be emitted, depending on the options selected during system configuration.

Log Records, although different in format, length, and content, always have the same heading:

Field Length (Words)	Contents	
(1)	RECORD TYPE	
(1)	RECORD LENGTH IN WORDS	
(3)	TIME STAMP (as defined in the MPE/3000 Intrinsic, CHRONOS)	
(1)	JOB TYPE	JOB NUMBER
Bits	0	2 15

In this format, the following definitions apply:

RECORD TYPE defines the record as one of the seven types described above.

RECORD LENGTH defines the number of words that the record contains.

TIME STAMP is the following three-word quantity:

Bits 0 6 7 15

Year (last 2 digits)	Day of Year
Hours	Minutes
Seconds	Tenth-of-Second

Bits 0 7 8 15

JOB TYPE is the type of main process being run, where

Bits 0:2	Have the Meaning
(00)	System
(01)	Session
(10)	Job

JOB NUMBER is a number defining the job/session under which the Log Record has been output. If this last word is 0, this means that the record is related to the system and was not output for a user.

Beyond the heading, the information in each individual record depends on the type of record and the particular event logged.

The general formats of all seven types of records are described below.

Log Error Record (Type 0)

This record is issued after a recoverable logging error has occurred, and when logging resumes. Its format is:

Field Length (words)	Contents	Values Always Taken
(1)	RECORD TYPE	= 0
(1)	RECORD LENGTH	= 10
(3)	TIME STAMP	
(1)	JOB TYPE/JOB NUMBER	= 0
(2)	MISSING LOG RECORDS	
(1)	MISSING JOB INITIATIONS	
(1)	MISSING JOB TERMINATIONS	

In this record, the following definitions apply:

JOB TYPE/JOB NUMBER is always set to 0.

MISSING LOG RECORDS (2 words) is the total number of Log Record events occurring while the Logging System was suspended.

MISSING JOB INITIATIONS (1 word) is the number of jobs/sessions initiated while logging was suspended.

MISSING JOB TERMINATIONS (1 word) is the number of jobs/sessions terminated while logging was suspended.

Note: The last three numbers correspond to Log Records that would have been emitted if logging was not suspended.

Head Record (Type 1)

This record is issued after each cold-load or reload before the system is up. Its format is:

	Field Length (words)	Contents	Values Always Taken
	(1)	RECORD TYPE	= 1
	(1)	RECORD LENGTH	= 17
	(3)	TIME STAMP	
	(1)	JOB TYPE/JOB NUMBER	= 0
System Number	(1)	UPDATE LEVEL	
	(1)	FIX LEVEL	
	(1)	CORE SIZE	
	(1)	AVAILABLE CST SIZE	
	(1)	AVAILABLE DST SIZE	
	(1)	AVAILABLE PCB SIZE	
	(1)	IOQ SIZE	
	(1)	TRL SIZE	
	(1)	ICS SIZE	
	(1)	MAX # of JOBS	
	(1)	MAX # of RUNNING JOBS	

The field definitions are:

JOB TYPE/JOB NUMBER is always set to 0.

SYSTEM NUMBER (2 words) is comprised of:

UPDATE LEVEL (1 word) is the update level of the system, composed of two ASCII characters.

FIX LEVEL (1 word) is the fix level of the system, composed of two ASCII characters.

CORE SIZE (1 word) is the main-memory size in K (1024) words of memory.

AVAILABLE CST SIZE (1 word) is the number of entries in the Code Segment Table.

AVAILABLE DST SIZE (1 word) is the number of entries in the Data Segment Table.

AVAILABLE PCB SIZE (1 word) is the number of entries in the Process Control Block.

IOQ SIZE (1 word) is the number of entries in the Input/Output Queue.

TRL SIZE (1 word) is the number of entries in the Time Request List.

ICS SIZE (1 word) is the number of words in the Interrupt Control Stack.

MAX # OF JOBS (1 word) is the maximum number of jobs/sessions allowed in the system.

MAX # OF RUNNING JOBS (1 word) is the maximum number of running jobs/sessions allowed in execution.

Job Initiation Record (Type 2)

This record is issued by the MPE/3000 Command Interpreter following a successful log-on to MPE/3000. Its format is:

Field Length (words)	Contents	Values Always Taken
(1)	RECORD TYPE	= 2
(1)	RECORD LENGTH	= 29
(3)	TIME STAMP	
(1)	JOB TYPE JOB NUMBER	
(4)	USER NAME	
(4)	ACCOUNT NAME	
(4)	JOB NAME	
(4)	LOGON GROUP NAME	
(1)	INPUT LOGICAL DEVICE NUMBER	
(1)	OUTPUT LOGICAL DEVICE NUMBER	
(2)	reserved	
(2)	CPU TIME LIMIT	
(1)	INPRI reserved	

The field definitions are:

JOB TYPE/JOB NUMBER (1 word) are the type and number of job/session.

USER NAME (4-word array) is the user name as specified on the :JOB card or in the :HELLO command, left-justified, blank-padded.

ACCOUNT NAME (4-word array) is the account name as specified on the :JOB card or in the :HELLO command, left-justified, blank-padded.

JOB NAME (4-word array) is the job name as specified on the :JOB card, or the session name as specified in the :HELLO command, left-justified, blank-padded. If *jobname* or *sessionname* was omitted, the array is filled with blanks.

LOGON GROUP NAME (4-word array) is the group name under which the log-on was performed, left-justified, blank-padded.

INPUT LOGICAL DEVICE NUMBER (1 word) is the logical device number of the standard input device for the job/ session.

OUTPUT LOGICAL DEVICE NUMBER (1 word) is the logical device number of the standard listing device for the job/ session.

CPU TIME LIMIT (2 words) is a double-word showing the central processor time limit as specified on the :JOB card or in the :HELLO command. If no limit applied, the field contains -1. If omitted, the field contains 0.

INPRI (1 byte, left-justified) is the job selection priority as defined on the :JOB card.

Job Termination Record (Type 3)

This record is issued at the end of a job or session, regardless of the cause of termination. Its format is:

Field Length (words)	Contents	Values Always Taken
(1)	• RECORD TYPE	= 3
(1)	RECORD LENGTH	= 12
(3)	TIME STAMP	
(1)	JOB TYPE JOB NUMBER	
(1)	MAXIMUM PRIORITY	
(1)	NUMBER OF CREATIONS	
(2)	CPU TIME IN SECONDS	
(2)	ELAPSED TIME IN MINUTES	

The field definitions are:

JOB TYPE/JOB NUMBER (1 word) are the type and number of the job/session.

MAXIMUM PRIORITY (1 word) is the lowest priority number (highest priority) ever run under by any process of the job/session. This value can range from 0 to 255.

NUMBER OF PROCESSES CREATED (1 word) is the total number of processes created under the main process during the job/session.

CPU TIME (2 words) is a double-word containing the total central processor time used (in seconds) by all processes of the job/session.

ELAPSED TIME (2 words) is a double-word containing the wall clock time (in minutes) during which the job/session has existed in the system. For a session, this time is called *connect time.*

Process Termination Record (Type 4)

This record is issued when a user process other than a Main Process terminates. Its format is:

Field Length (words)	Contents	Values Always Taken
(1)	RECORD TYPE	= 4
(1)	RECORD LENGTH	= 11
(3)	TIME STAMP	
(1)	JOB TYPE JOB NUMBER	
(1)	# OF PROGRAM FILE SEGMENTS	
(1)	# OF SL SEGMENTS (non MPE)	
(1)	MAXIMUM STACK SIZE EVER	
(1)	MAXIMUM DATA SEGMENT SIZE EVER	
(1)	CUMULATIVE TOTAL OF VIRTUAL STORAGE	

The field definitions are:

JOB TYPE/JOB NUMBER (1 word) are the type and number of the job/session.

NUMBER OF PROGRAM FILE SEGMENTS (1 word) is the number of segments contained in the program file loaded on behalf of the process.

NUMBER OF NON-MPE SL SEGMENTS (1 word) is the number of segments from the segmented library (excluding MPE/3000), loaded on behalf of the process.

MAXIMUM STACK SIZE EVER (1 word) is the largest size (in words) ever attained by the stack during process life.

MAXIMUM DATA SEGMENT SIZE EVER (1 word) is the largest size (in words) ever attained by an extra data segment during process life.

TOTAL AMOUNT OF VIRTUAL STORAGE REQUESTED (1 word) is the total amount of disc space (in sectors) requested for data (stack and extra data segments) during process life.

File Close Record (Type 5)

This record is issued whenever a file is closed. Its format is:

Field Length (words)	Contents		Values Always Taken
(1)	RECORD TYPE		= 5
(1)	RECORD LENGTH		= 28
(3)	TIME STAMP		
(1)	JOB TYPE	JOB NUMBER	
(14)	FILENAME	(Reserved)	
(1)	DISPOSITION	DOMAIN	
(2)	# of SECTORS ALLOCATED		
(1)	DEVICE TYPE	DEVICE NUMBER	
(2)	# of RECORDS PROCESSED		
(2)	# of BLOCKS PROCESSED		

The field definitions are:

JOB TYPE/JOB NUMBER (1 word) are the type and number of the job/session.

FULL FILENAME (27 bytes) contains the *filename.groupname.accountname* of the file; each name is eight bytes long, with the last byte containing a blank.

DISPOSITION (1 byte) is the file disposition as specified in the FCLOSE intrinsic.

DOMAIN (1 byte) is the file domain, as specified in the FOPEN intrinsic.

NUMBER OF SECTORS ALLOCATED (2 words) is the physical space (in sectors) actually reserved on disc for the file. (When the file does not reside on disc, this value is 0.)

DEVICE TYPE (1 byte) is the device type of the device on which the file resides, as returned by the FGETINFO intrinsic.

DEVICE NUMBER (1 byte) is the logical device number of the device on which the file resides.

RECORDS PROCESSED (2 words) is the number of records processed since the last FOPEN on that file by the process.

BLOCKS PROCESSED (2 words) is the number of blocks written and read to and from the file since the last FOPEN on the file by the process.

Shut-Down Record (Type 6)

This record is output when the system is shut down (by the =SHUTDOWN console command). Its format is:

Field Length (words)	Contents	Values Always Taken
(1)	RECORD TYPE	= 7
(1)	RECORD LENGTH	= 9
(3)	TIME STAMP	
(1)	JOB TYPE JOB NUMBER	= 0
(1)	# of JOBS	
(1)	# of SESSIONS	
(1)	(reserved)	= 0

The field definitions are:

JOB TYPE/JOB NUMBER is always set to 0.

NUMBER OF JOBS ON SYSTEM (1 word) is the number of jobs on the system when the command took effect.

NUMBER OF SESSIONS ON SYSTEM (1 word) is the number of sessions on the system when the command took effect.

LOG FILE ERRORS

The conditions that can cause the Logging System to fail are denoted by the error numbers described below. (In this list, *R* indicates a recoverable error while *I* denotes an irrecoverable one.) These error numbers are reported to the user through various console error messages described in the next sub-section.

Error No.	Error	Recoverable/ Irrecoverable
1	Input/output error in accessing the system disc.	I
2	Input/output error in accessing disc Log File.	I
21	Data parity error.	I
26	Transmission error.	I
27	Input/output time-out.	I
28	Time-up error or data overrun.	I
29	SIO failure.	I
30	Unit failure	I
46	Insufficient disc space to create Log File.	R
47	Input/output error on file lable.	I
57	Virtual memory not sufficient.	I
61	Group (PUB) disc space exceeded in creating Log File.	R
62	Account (SYS) disc space exceeded in creating Log File.	R
63	Group disc space exceeded in allocating new extent to the Log File.	R
64	Account disc space exceeded in allocating new extent to the Log File.	R
100	A file of the same name as the current Log File already exists in the system file directory.	R
102	Directory input/output error.	I
103	System directory overflow.	I

CONSOLE MESSAGES FOR LOG FILES

Log File errors and status are reported to the system console through messages of the following format:

ST/hh:mm/message

hh	The hour of the day.
mm	The minute of the hour.
message	The message text.

The message text may consist of any of the following messages:

LOG FILE NUMBER x ON

A new Log File (number *x*) has been created. This message always appears before the WELCOME message after cold-load. When this message appears while the system is running, it indicates that the previous current Log File has been closed. (The actual file name of the Log File is *LOGxxxx*, where *xxxx* are four characters representing the same value as *x* in the above message, with leading zeros.)

LOG FILE NUMBER x IS 1/2 FULL

The total space now occupied by the Log File data is half the allotted file size.

LOG FILE NUMBER x IS 3/4 FULL

The total space now occupied by the Log File data is 3/4 the allotted file size.

LOG FILE NUMBER x ERROR #nn. LOGGING STOPPED

A *fatal* Log File error (described by the error number under *Log File Errors*) occurred; *nn* is the error number. Logging is disabled until the next cold load or reload.

LOG FILE NUMBER x ERROR #nn. LOGGING SUSPENDED

A recoverable Log File error (described by the error number under *Log File Errors*) occurred; *nn* is the error number. Logging is temporarily suspended pending a :RESUMELOG Command.

LOG FILE NUMBER x. LOGGING RESUMED

A :RESUMELOG Command was successfully executed.

APPENDIX A
ASCII Character Set

Character	Octal Value (Left Byte)	Octal Value (Right Byte)	Meaning	
NUL	000000	000000	Null (Normally ignored by MPE/3000)	
SOH	000400	000001	Start of heading	
STX	001000	000002	Start of text	
ETX	001400	000003	End of text	
EOT	002000	000004	End of transmission	
ENQ	002400	000005	Enquiry	
ACK	003000	000006	Acknowledge	
BEL	003400	000007	Bell	
BS	004000	000010	Backspace	
HT	004400	000011	Horizontal tabulation	
LF	005000	000012	Line feed (Normally ignored by MPE/3000)	
VT	005400	000013	Vertical tabulation	
FF	006000	000014	Form feed	
CR	006400	000015	Carriage return	
SO	007000	000016	Shift out	
SI	007400	000017	Shift in	
DLE	010000	000020	Data link escape	(Normally ignored by MPE/3000)
DC1	010400	000021	Device control 1	
DC2	011000	000022	Device control 2	
DC3	011400	000023	Device control 3	
DC4	012000	000024	Device control 4	
NAK	012400	000025	Negative acknowledge	
SYN	013000	000026	Synchronous idle	
ETB	013400	000027	End of transmission block	

Character	Octal Value (Left Byte)	Octal Value (Right Byte)	Meaning
CAN	014000	000030	Cancel
EM	014400	000031	End of medium
SUB	015000	000032	Substitute
ESC	015400	000033	Escape
FS	016000	000034	File separator
GS	016400	000035	Group separator
RS	017000	000036	Record separator
US	017400	000037	Unit separator
SP	020000	000040	Space
!	020400	000041	Exclamation point
”	021000	000042	Quotation mark
#	021400	000043	Number sign
$	022000	000044	Dollar sign
%	022400	000045	Percent sign
&	023000	000046	Ampersand
’	023400	000047	Apostrophe
(	024000	000050	Opening parenthesis
)	024400	000051	Closing parenthesis
*	025000	000052	Asterisk
+	025400	000053	Plus
,	026000	000054	Comma
-	026400	000055	Hyphen (Minus)
.	027000	000056	Period (Decimal)
/	027400	000057	Slant
0	030000	000060	Zero
1	030400	000061	One
2	031000	000062	Two
3	031400	000063	Three
4	032000	000064	Four
5	032400	000065	Five
6	033000	000066	Six
7	033400	000067	Seven

Character	Octal Value (Left Byte)	Octal Value (Right Byte)	Meaning
8	034000	000070	Eight
9	034400	000071	Nine
:	035000	000072	Colon
;	035400	000073	Semi-colon
<	036000	000074	Less than
=	036400	000075	Equals
>	037000	000076	Greater than
?	037400	000077	Question mark
@	040000	000100	Commercial at
A	040400	000101	Uppercase A
B	041000	000102	Uppercase B
C	041400	000103	Uppercase C
D	042000	000104	Uppercase D
E	042400	000105	Uppercase E
F	043000	000106	Uppercase F
G	043400	000107	Uppercase G
H	044000	000110	Uppercase H
I	044400	000111	Uppercase I
J	045000	000112	Uppercase J
K	045400	000113	Uppercase K
L	046000	000114	Uppercase L
M	046400	000115	Uppercase M
N	047000	000116	Uppercase N
O	047400	000117	Uppercase O
P	050000	000120	Uppercase P
Q	050400	000121	Uppercase Q
R	051000	000122	Uppercase R
S	051400	000123	Uppercase S
T	052000	000124	Uppercase T
U	052400	000125	Uppercase U
V	053000	000126	Uppercase V
W	053400	000127	Uppercase W

Character	Octal Value (Left Byte)	Octal Value (Right Byte)	Meaning
X	054000	000130	Uppercase X
Y	054400	000131	Uppercase Y
Z	055000	000132	Uppercase Z
[	055400	000133	Opening bracket
\	056000	000134	Reverse slant
]	056400	000135	Closing bracket
^	057000	000136	Circumflex
_	057400	000137	Underscore
`	060000	000140	Grave accent
a	060400	000141	Lowercase a
b	061000	000142	Lowercase b
c	061400	000143	Lowercase c
d	062000	000144	Lowercase d
e	062400	000145	Lowercase e
f	063000	000146	Lowercase f
g	063400	000147	Lowercase g
h	064000	000150	Lowercase h
i	064400	000151	Lowercase i
j	065000	000152	Lowercase j
k	065400	000153	Lowercase k
l	066000	000154	Lowercase l
m	066400	000155	Lowercase m
n	067000	000156	Lowercase n
o	067400	000157	Lowercase o
p	070000	000160	Lowercase p
q	070400	000161	Lowercase q
r	071000	000162	Lowercase r
s	071400	000163	Lowercase s
t	072000	000164	Lowercase t
u	072400	000165	Lowercase u
v	073000	000166	Lowercase v
w	073400	000167	Lowercase w

Character	Octal Value (Left Byte)	Octal Value (Right Byte)	Meaning
x	074000	000170	Lowercase x
y	074400	000171	Lowercase y
z	075000	000172	Lowercase z
{	075400	000173	Opening (left) brace
|	076000	000174	Vertical line
}	076400	000175	Closing (right) brace
~	077000	000176	Tilde
DEL	077400	000177	Delete

APPENDIX B
Summary of Commands

SYSTEM MANAGER CAPABILITY COMMANDS

Commands	Parameters	Function	When Issued	Text Discussion
:ALTACCT	*acctname* *[;PASS=[password]]* *[;FILES=[filespace]]* *[;CPU=[cpu]]* *[;CONNECT=[connect]]* *[;CAP=[capabilitylist]]* *[;ACCESS=fileaccess]]* *[;MAXPRI=[subqueuename]]* *[;LOCATTR=[localattribute]]*	Changes an account's characteristics.	Job, session, break, or programmatically.	6-7
:LISTACCT	*[@ accountname] [,listfile]*	Lists attributes of an account.	Job, session, break, or programmatically.	6-8
:LISTGROUP	*[groupset] [,listfile]*	Lists attributes of a group.	Job, session, break, or programmatically.	6-14
:LISTUSER	*[userset] [,listfile]*	Lists attributes of a user.	Job, session, break, or programmatically.	6-19
:NEWACCT	*acctname,mgrname* *[;PASS=[password]]* *[;FILES=[filespace]]* *[;CPU=[cpu]]* *[;CONNECT=[connect]]* *[;CAP=[capabilitylist]]* *[;ACCESS=[fileaccess]]* *[;MAXPRI=[subqueuename]]* *[;LOCATTR=[localattribute]]*	Creates a new account.	Job, session, break, or programmatically.	6-4, 6-5
:PURGEACCT	*acctname*	Deletes an account.	Job, session, break, or programmatically.	6-11

Commands	Parameters	Function	When Issued	Text Discussion
:REPORT	*[groupset]* *[,listfile]*	Displays an account's resource usage.	Job, session, break, or programmatically.	7-2
:RESETACCT	*[@* *acctname]* *[,CPU* *,CONNECT]*	Resets resource-use counters for an account and its groups.	Job, session, break, or programmatically.	7-3

ACCOUNT MANAGER CAPABILITY COMMANDS

Commands	Parameters	Function	When Issued	Text Discussion
:ALTGROUP	*groupname* *[;PASS=[password]]* *[;CAP=[capabilitylist]]* *[;FILES=[filespace]]* *[;CPU=[cpu]]* *[;CONNECT=[connect]]* *[;ACCESS=[fileaccess]]*	Changes a group's attributes.	Job, session, break, or programmatically.	6-13
:ALTUSER	*username* *[;PASS=[password]]* *[;CAP=[capabilitylist]]* *[;MAXPRI=[subqueuename]]* *[;LOCATTR=[localattribute]]* *[;HOME=[homegroupname]]*	Changes a user's attributes.	Job, session, break, or programmatically.	6-19
:LISTACCT	*[acctname] [,listfile]*	Lists attributes of user's log-on account.	Job, session, break, or programmatically.	6-8
:LISTGROUP	*[groupset] [,listfile]*	List attributes of a group in user's log-on account.	Job, session, break, or programmatically.	6-14
:LISTUSER	*[userset] [,listfile]*	Lists attributes of a user in log-on account.	Job, session, break, or programmatically.	6-19
:NEWGROUP	*groupname* *[;PASS=[password]]* *[;CAP=[capabilitylist]]* *[;FILES=[filespace]]* *[;CPU=[cpu]]* *[;CONNECT=[connect]]* *[;ACCESS=[fileaccess]]*	Creates a new group in log-on account.	Job, session, break, or programmatically.	6-12

Commands	Parameters	Function	When Issued	Text Discussion
:NEWUSER	*username* *[;PASS=[password]]* *[;CAP=[capabilitylist]]* *[;MAXPRI=[subqueuename]]* *[;LOCATTR=[localattribute]]* *[;HOME=[homegroupname]]*	Creates a new user in log-on account.	Job, session break, or programmatically.	6-17
:PURGEGROUP	*groupname*	Deletes a group from log-on account.	Job, session, break, or programmatically.	6-17
:PURGEUSER	*username*	Deletes a user from log-on account.	Job, session, break, or programmatically.	6-22
:REPORT	*[groupset] [,listfile]*	Displays resource-usage counts for log-on account and its groups.	Job, session, break, or programmatically.	7-2

SYSTEM SUPERVISOR CAPABILITY COMMANDS

Commands	Parameters	Function	When Issued	Text Discussion
:ALLOCATE	*[code],name*	Permanently allocates a program or procedure in virtual memory.	Job or session.	8-12
:DEALLOCATE	*[code],name*	Removes a program or procedure from virtual memory.	Job, session, break, or programmatically.	8-13
:QUANTUM	*subqueuename,time*	Changes a circular subqueue time-quantum.	Job, session, break, or programmatically.	8-4
:RESUMELOG		Resumes logging following suspension caused by an error.	Job, session, break, or programmatically.	9-5
:SHOWLOG		Displays Log-File status.	Job, session, break, or programmatically.	9-4
:SHOWQ	*[subqueuename]*	Displays scheduling subqueue information.	Job, session, break, or programmatically.	8-5

Commands	Parameters	Function	When Issued	Text Discussion
:SWITCHLOG		Closes current Log File, opens a new Log File, and treats this as current Log File.	Job, session, break, or programmatically.	9-5
:SYSDUMP	*dumpfile[,auxlistfile]*	Copies MPE/3000 to tape.	Job or session.	4-8, 4-9

APPENDIX C
Summary of Intrinsic Calls

The two intrinsic calls available to the user with System Supervisor Capability are summarized below. For each intrinsic, the complete declaration head appears; the intrinsic call format is distinguished from the remainder of the head by a box. The function of the intrinsic is described. Optional parameters are *bold face* in the intrinsic head. All condition codes that can be returned by the intrinsic are listed.

The intrinsic error-code numbers are also presented. (These are the numbers that appear in the abort-error messages generated when errors are encountered in the corresponding intrinsics.) The functional categories represented by these error numbers are

Error Number Range	Functional Category
1—29	File Management
30—39	Resource Management
40—49	System Timer (Clock)
50—59	Traps
60—79	Utility Routines
80—99	Program Management
100—119	Process Control
120—129	Scheduling
130—149	Data Segments
180—199	Input/Output Utilities
200—209	Special Utilities

Where intrinsics have special attributes, those attributes are noted as follows:

- Uncallable intrinsics (those that can only be invoked by the user in privileged mode) are indicated by an asterisk (*).

- Intrinsics that can be called in privileged mode even though the user does not have the appropriate capability are denoted by a plus sign (+).

- Intrinsics that can be called by a process when the DB register is not pointing to that process' stack are denoted by a dollar sign ($).

PROCESS-HANDLING OPTIONAL CAPABILITY INTRINSICS

PROCEDURE *CREATE* *(progname*, **entryname**, *pin*, **param**, **flags**, **stacksize**, **dlsize**, **maxdata**, **priorityclass**, **rank**);

VALUE **param,stacksize,dlsize,priorityclass,maxdata,flags,rank;**

LOGICAL **priorityclass,flags;**

INTEGER **stacksize,dlsize,maxdata,***pin*,**param,rank;**

BYTE ARRAY *progname*,**entryname;**

OPTION VARIABLE, EXTERNAL;

FUNCTION:	*Creates a process.*
TYPE:	*None*
OPTIONAL PARAMETERS:	**entryname,param,flags,stacksize,dlsize,maxdata,priorityclass, rank**
CONDITION CODES:	*CCE, CCG, CCL*
ERROR CODE:	*100*
SPECIAL ATTRIBUTES:	*None*
TEXT DISCUSSION:	*Page 8-7*

```
PROCEDURE               GETPRIORITY (pin, priorityclass, rank);

VALUE        pin, priorityclass,rank

LOGICAL      priorityclass;

INTEGER      rank

OPTION VARIABLE, EXTERNAL;
```

FUNCTION:	*Reschedules a process.*
TYPE:	*None*
OPTIONAL PARAMETERS:	rank
CONDITION CODES:	*CCE, CCL, CCG*
ERROR CODE:	*120*
SPECIAL ATTRIBUTES:	*$*
OPTIONAL CAPABILITIES:	*Process Handling*
TEXT DISCUSSION:	*Page 8-11*

APPENDIX D

Driver Names, Types, Subtypes, and Sizes

Device	Part No.	Driver Name	Type	Sub-Type	Record Width (decimal words)
System Clock/ Console Interface	30031A	IOCLTTY0	16		36
ASR 33,35	30124A			0	
Terminet 10 cps	30120A			1	
Terminet 15 cps				2	
Terminet 30 cps				3	
HP 2600A, 10 cps	30123A			4	
HP 2600A, 15 cps				5	
HP 2600A, 30 cps				6	
HP 2600A, 60 cps				7	
HP 2600A, 120 cps				8	
HP 2600A, 240 cps				9	
Asynchronous Terminal Controller	30032A	IOTERM0	16		36
Hardwired Terminal				0*	
Terminal Interfaced Over 103A modem.				1	

*These terminals should be configured with Sub-Type = 1 when hardwired: ASR 37, IBM Selectric, Memorex 1240.

Device	Part No.	Driver Name	Type	Sub-Type	Record Width (decimal words)
Terminal Interfaced Over 202 modem				2	
Terminal Interfaced Over 2002 modem (European)				3	
Nine-channel Magnetic Tape Unit	30115A-100	IOTAPE0	24	0	128
Fixed-Head Disc	30103A	IOFDISK0	1		128
1 megabyte				0	
2 megabyte	-001			1	
4 megabyte	-002			2	
Cartridge Disc (7900)	30110A	IOMDISK0	0	2	128
Disc File	30102A	IOMDISK0	0	3	128
Card Reader	30106A, 30107A	IOCDRD0	8	0	40
Line Printer	30108A-001 30109A-001	IOLPRT0	32	0	66
Paper Tape Reader	30104A	IOPTRD0	9	0	128
Paper Tape Punch	30105A	IOPTPN0	34	0	128
Plotter	30226A	IOPLOT0	35	0	128
Card Punch	30112A	IOCDPN0	33	0	40

APPENDIX E
Default Parameter Settings for Configuration

Memory Size						
Configurator Question	32	48	64	Unit of Measure	Minimum Allowed	Maximum Allowed
CST = xxx.?	192	256	256	entries	16	256
DST = xxx.?	96	128	192	entries	1	1024
PCB = xxx.?	44	56	80	entries	2	256
I/O QUEUE = xxx.?	24	48	64	entries	1	255
TERMINAL BUFFERS = xxx.?	12	24	48	buffers	1	255
IOCB = xxx.?	16	32	48	entries	1	255
ICS = xx.?	256	384	512	words	128	1024
UCOP REQUEST QUEUE = xx?	8	16	32	entries	1	256
TIMER REQUEST LIST = xx.?	16	32	48	– –	1	128
NO. OF SECONDS TO LOGON = xxx.?	120	120	120	seconds	10	600
NO. OF RINS = xxx.?	16	32	64	– –	5	1024
MAX # OF GLOBAL RINS = xxx.?	8	16	32	– –	0	1024
MAX # OF JOBS = xxx.?	6	10	20	– –	1	255
MAX # OF CONCURRENT RUNNING JOBS = xxx.?	6	10	20	– –	1	255
DEFAULT JOB CPU TIME LIMIT = xxx?	0	0	0	– –	0	32767
LOG FILE RECORD SIZE (SECTORS) = x?	2	2	2	– –	1	8
LOG FILE SIZE (RECORDS) = xxxx?	1024	1024	1024	– –	16	32767
VIRTUAL MEMORY = xxxx.?	1024	2048	3072	sectors	1024	8192
DIRECTORY = yyyy, MIN = zzzz, MAX = xxxx?	256	384	512	sectors (max)	3	6000
TIME-QUANTUM – CS SUBQUEUE = xxx.?	1000	1000	1000	milliseconds	20	32767
TIME-QUANTUM – DS SUBQUEUE = xxx.?	1000	1000	1000	milliseconds	20	32767
MAX CODE SEGMENTS SIZE = xxxx.?	4096	4096	4096	words	1024	16384
MAX # OF CODE SEGMENTS/PROCESS = xxx.?	40	40	40	– –	1	255
MAX STACK SIZE = xxxxx.?	12288	24576	31232	words	256	31232
MAX EXTRA DATA SEG SIZE = xxxx.?	2048	3072	4096	words	0	32767
MAX # OF EXTRA DATA SEGMENTS/PROCESS = x.?	2	3	4	– –	0	4
STD STACK SIZE = xxxx.?	800	800	800	words	256	4096

APPENDIX F
Device and Driver Information for Configuration

For the following HP-supported devices and drivers, the user should input the information described below in Steps 3.3.3 through 3.3.11 and Step 10.1 of the Configurator/User Dialogue.

SYSTEM CLOCK/CONSOLE INTERFACE DRIVER (IOCLTTY0)

Step No.	Configurator Output	User Response
3.3.3	TYPE?	16
3.3.4	SUB-TYPE?	(A digit, 0 through 9 — see Appendix D)
3.3.5	REC WIDTH?	36
3.3.6	OUTPUT DEVICE?	0
3.3.7	ACCEPT JOBS/SESSIONS?	YES
3.3.8	ACCEPT DATA?	NO
3.3.9	INTERACTIVE?	YES
3.3.10	DUPLICATIVE?	YES
3.3.11	DRIVER NAME?	IOCLTTY0
10.1	ENTER PROGRAM NAME, REPLACEMENT FILE NAME?	IOCLTTY0, <prepared file name>

ASYNCHRONOUS TERMINAL CONTROLLER DRIVER (IOTERM0)

Step No.	Configurator Output	User Response
3.3.3	TYPE?	16
3.3.4	SUB-TYPE?	(0, 1, 2, or 3 — see Appendix D).
3.3.5	REC WIDTH?	(Varies with terminal)
3.3.6	OUTPUT DEVICE?	(No. of corresponding listing device)
3.3.7	ACCEPT JOBS/SESSIONS?	YES
3.3.8	ACCEPT DATA?	NO
3.3.9	INTERACTIVE?	YES
3.3.10	DUPLICATIVE?	YES
3.3.11	DRIVER NAME?	IOTERM0
10.1	ENTER PROGRAM NAME, REPLACEMENT FILE NAME?	IOTERM0, <prepared file name>

NINE-CHANNEL MAGNETIC TAPE UNIT DRIVER (IOTAPE0)

Step No.	Configurator Output	User Response
3.3.3	TYPE?	24
3.3.4	SUB-TYPE?	0
3.3.5	REC WIDTH?	128
3.3.6	OUTPUT DEVICE?	0
3.3.7	ACCEPT JOBS/SESSIONS?	YES
3.3.8	ACCEPT DATA?	NO
3.3.9	INTERACTIVE?	NO
3.3.10	DUPLICATIVE?	NO
3.3.11	DRIVER NAME?	IOTAPE0
10.1	ENTER PROGRAM NAME, REPLACEMENT FILE NAME?	IOTAPE0, <prepared file name>

FIXED-HEAD DISC DRIVER (IOFDISK0)

Step No.	Configurator Output	User Response
3.3.3	TYPE?	1
3.3.4	SUB-TYPE?	(0, 1, or 2)
3.3.5	REC WIDTH?	128
3.3.6	OUTPUT DEVICE?	0
3.3.7	ACCEPT JOBS/SESSIONS?	NO
3.3.8	ACCEPT DATA?	NO
3.3.9	INTERACTIVE?	NO
3.3.10	DUPLICATIVE?	NO
3.3.11	DRIVER NAME?	IOFDISK0
10.1	ENTER PROGRAM NAME, REPLACEMENT FILE NAME?	IOFDISK0, <prepared file name>

CARTRIDGE DISK DRIVER (IOMDISK0)

Step No.	Configurator Output	User Response
3.3.3	TYPE?	0
3.3.4	SUB-TYPE?	(0 or 2)
3.3.5	REC WIDTH?	128
3.3.6	OUTPUT DEVICE?	0
3.3.7	ACCEPT JOBS/SESSIONS?	NO
3.3.8	ACCEPT DATA?	NO
3.3.9	INTERACTIVE?	NO
3.3.10	DUPLICATIVE?	NO
3.3.11	DRIVER NAME?	IOMDISK0
10.1	ENTER PROGRAM NAME, REPLACEMENT FILE NAME?	IOMDISK0, <prepared file name>

DISC FILE DRIVER (IOMDISK0)

Step No.	Configurator Output	User Response
3.3.3	TYPE?	0
3.3.4	SUB-TYPE?	3
3.3.5	REC WIDTH?	128
3.3.6	OUTPUT DEVICE?	0
3.3.7	ACCEPT JOBS/SESSIONS?	NO
3.3.8	ACCEPT DATA?	NO
3.3.9	INTERACTIVE?	NO
3.3.10	DUPLICATIVE?	NO
3.3.11	DRIVER NAME?	IOMDISK0
10.1	ENTER PROGRAM NAME, REPLACEMENT FILE NAME?	IOMDISK0 <prepared file name>

CARD READER DRIVER (IOCDR0)

Step No.	Configurator Output	User Response
3.3.3	TYPE?	8
3.3.4	SUB-TYPE?	0
3.3.5	REC WIDTH?	40
3.3.6	OUTPUT DEVICE?	(Device desired)
3.3.7	ACCEPT JOBS/SESSIONS?	YES
3.3.8	ACCEPT DATA?	YES
3.3.9	INTERACTIVE?	NO
3.3.10	DUPLICATIVE?	NO
3.3.11	DRIVER NAME?	IOCDRD0
10.1	ENTER PROGRAM NAME, REPLACEMENT FILE NAME?	IOCDRD0, < prepared file name>

LINE PRINTER DRIVER? (IOLPRT0)

Step No.	Configurator Output	User Response
3.3.3	TYPE?	32
3.3.4	SUB-TYPE?	0
3.3.5	REC WIDTH?	66
3.3.6	OUTPUT DEVICE?	0
3.3.7	ACCEPT JOBS/SESSIONS?	NO
3.3.8	ACCEPT DATA?	NO
3.3.9	INTERACTIVE?	NO
3.3.10	DUPLICATIVE?	NO
3.3.11	DRIVER NAME?	IOLPRT0
10.1	ENTER PROGRAM NAME, REPLACEMENT FILE NAME?	IOLPRT0, <prepared file name>

PAPER TAPE READER DRIVER (IOTRD0)

Step No.	Configurator Output	User Response
3.3.3	TYPE?	9
3.3.4	SUB-TYPE?	0
3.3.5	REC WIDTH?	128
3.3.6	OUTPUT DEVICE?	0
3.3.7	ACCEPT JOBS/SESSIONS?	NO
3.3.8	ACCEPT DATA?	NO
3.3.9	INTERACTIVE?	NO
3.3.10	DUPLICATIVE?	NO
3.3.11	DRIVER NAME?	IOPTRD0
10.1	ENTER PROGRAM NAME, REPLACEMENT FILE NAME?	IOPTRD0, <prepared file name>

PAPER TAPE PUNCH DRIVER (IOPTPN0)

Step No.	Configurator Output	User Response
3.3.3	TYPE?	34
3.3.4	SUB-TYPE?	0
3.3.5	REC WIDTH?	128
3.3.6	OUTPUT DEVICE?	0
3.3.7	ACCEPT JOBS/SESSIONS?	NO
3.3.8	ACCEPT DATA?	NO
3.3.9	INTERACTIVE?	NO
3.3.10	DUPLICATIVE?	NO
3.3.11	DRIVER NAME?	IOPTPN0
10.1	ENTER PROGRAM NAME, REPLACEMENT FILE NAME?	IOPTPN0, <prepared program file name>

PLOTTER DRIVER (IOPLOT0)

Step No.	Configurator Output	User Response
3.3.3	TYPE?	35
3.3.4	SUB-TYPE?	0
3.3.5	REC WIDTH	255
3.3.6	OUTPUT DEVICE?	0
3.3.7	ACCEPT JOBS/SESSIONS?	NO
3.3.8	ACCEPT DATA?	NO
3.3.9	INTERACTIVE?	NO
3.3.10	DUPLICATIVE?	NO
3.3.11	DRIVER NAME?	IOPLOT0
10.1	ENTER PROGRAM NAME, REPLACEMENT FILE NAME?	IOPLOT0, <prepared program file name>

CARD PUNCH DRIVER (IOCDPN0)

Step No.	Configurator Output	User Response
3.3.3	TYPE?	33
3.3.4	SUB-TYPE?	0
3.3.5	REC WIDTH?	40
3.3.6	OUTPUT DEVICE?	0
3.3.7	ACCEPT JOBS/SESSIONS?	NO
3.3.8	ACCEPT DATA?	NO
3.3.9	INTERACTIVE?	NO
3.3.10	DUPLICATIVE?	NO
3.3.11	DRIVER NAME?	IOCDPN0
10.1	ENTER PROGRAM NAME, REPLACEMENT FILE NAME?	IOCDPN0, <prepared program file name>

APPENDIX G

File Back-Up and System Reload Using SYSDUMP Command

To ensure that data files are not inadvertently lost during a normal course of operation, a system back-up (page 4-27) and reload capability (page 5-2) has been made available. The system back-up facility provides the System Supervisor User with a copy of the operating system, the current directory and accounting information, and either all of the user files or only those user files which have most recently been changed. The RELOAD option of the cold-load operation allows the Console Operator to recreate the system, its directory, and user files as of the date on which the last :SYSDUMP operation was performed.

The following example illustrates the use of :SYSDUMP to back-up a system, and the operation required to re-establish the system as of the date of the last :SYSDUMP operation.

EXAMPLE

The :SYSDUMP command is used to dump the system, file directory, and all files on Sunday evening of each week. On subsequent days of the week, only those files that have changed since Sunday and the system and updated directory and accounting information are dumped. This procedure for a given week follows:

Date	Operation	Description
Sunday (9/9)	:FILE SUNDAY; DEV=TAPE :SYSDUMP *SUNDAY ANY CHANGES? NO ENTER DUMP DATE? 0 LIST FILES DUMPED? YES	*The system, file directory, and Files A, B, C, D, and E are saved on the tape file SUNDAY.*
Monday (9/10)	:FILE MONDAY; DEV=TAPE :SYSDUMP *MONDAY ANY CHANGES? NO ENTER DUMP DATE? 9/9/73 LIST FILES DUMPED? YES	*Files A and E were modified and F was added to the system. The dump provided on Monday includes the system, the file directory, and the Files A, E, and F.*

Date	Operation	Description
Tuesday (9/11)	:FILE TUESDAY; DEV=TAPE :SYSDUMP *TUESDAY ANY CHANGES? NO ENTER DUMP DATE? 9/9/73 LIST FILES DUMPED? YES	*Since Monday, none of the files were modified; however, File C was deleted from the system. This is reflected in the updated directory which is dumped. In addition, Files A, E, and F are again dumped, since they have been changed since Sunday.*
Wednesday (9/12)	— —	*On Wednesday morning, the files on the system were inadvertently lost.*

The system and its file domain are recreated as of its state on Tuesday by using the RELOAD option of the cold-load procedure, using the tape created by :SYSDUMP on Tuesday. When the message

```
NOT ALL FILES FOUND; ANOTHER TAPE SET AVAILABLE?
```

appears, the operator mounts Sunday's :SYSDUMP tape and replies YES to the question. This retrieves the rest of the files.

Another method of doing this, which would minimize the amount of time taken to do the dump each day, but would increase the time to reload, would be to enter Monday's date (9/10/73) when doing Tuesday's dump. In this case, no files would have been dumped since nothing has been changed since Monday. The directory on Tuesday's tape, however, would reflect the absence of the File C.

When the RELOAD operation was undertaken on Wednesday, Tuesday's tape would be used first. When another tape set was requested, Monday's tape would be mounted. When yet another tape set was requested, Sunday's tape would then be mounted, to complete the retrieval of files.

APPENDIX H
Back-up of Disc Files

MPE/3000 provides two methods of backing-up files. These are:

1. The :SYSDUMP command, used to dump all files on the system or only those files most recently changed, and the current directory and accounting information. (See also Section IV and Appendix G.)

2. The :STORE command, explained later in this appendix. This command is available to standard users in order to back up those files to which the user has read access. System Manager and System Supervisor Users, however, have the ability to dump any or all files in the system by using the :STORE command.

Tapes produced by :SYSDUMP and :STORE are compatible. Those written by :SYSDUMP are used by the MPE/3000 Initiator when reloading the system (RELOAD option, discussed in Section V). Tapes produced by either method are suitable as input to the :RESTORE command (described in detail later in this appendix) which allows a System Manager or System Supervisor User the ability to restore to disc from a back-up tape any file in the system.

In general, :SYSDUMP should be used for daily back-up of the system, since it provides a record of the latest accounting information. :RESTORE is used when it is desirable to back-up only those files which belong to a particular set of groups or accounts.

DUMPING FILES OFF-LINE

Any user with Standard MPE/3000 Capability can obtain a back-up copy of a particular user disc file or fileset, by copying the fileset off-line to a magnetic tape unit by issuing the :STORE command. The files are copied in a special format along with all descriptive information (such as account name, group name, and lockword), permitting them to be read back into the system later (by the :RESTORE command).

Multi-file reels and multi-reel files are both supported by :STORE and :RESTORE.

NOTE: The :STORE and :RESTORE commands are used primarily as a back-up for files. They can be used to interchange files between installations if the accounts, groups, and creators of the files to be restored are defined in the destination system. Furthermore, if no destination device is specified in the :RESTORE command, MPE/3000 does not guarantee which devices will actually receive the files — if a device of the same type as the original device with sufficient storage space cannot be found, the file is restored to any device that is a member of the device class DISC.

The :STORE command is written as follows:

:STORE [filesetlist] ;tapefile [;SHOW]

filesetlist A list of one or more files or filesets to be copied, written in this format:

[fileset[,fileset]...]

In this list, *fileset* must be written as one of the combinations noted below. If the entire *filesetlist* parameter is omitted, the default value is @ (all files of the log-on group are copied). If any file belonging to a fileset requires a lockword, the user may supply that lockword. If he fails to supply it (while programming in batch mode), the file will not be stored. If he fails to supply it (in interactive mode), he is prompted for the lockword. (Optional parameter.)

tapefile The name of the destination tape file onto which the stored files are written. This can be any magnetic tape file from the output set. This file must be referenced in the **formaldesignator* format. (Required parameter.)

SHOW A request that the names of the stored files be listed. If SHOW is omitted, only the total number of files stored, the names of the files not stored, and the number of files not stored, are listed. (Optional parameter.)

NOTE: Before issuing a :STORE command, the user must identify tapefile as a magnetic tape file. He does this through the :FILE command. In this case, the :FILE command should be written in the following format, including no parameters other than those shown:

:FILE formaldesignator { [=filereference] ,NEW } [DEV=device]

The device parameter must be the device class name or logical unit number of a magnetic tape unit.

All other parameters for tapefile are supplied by the :STORE command executor; if the user attempts to supply any of these himself, the :STORE command is rejected.

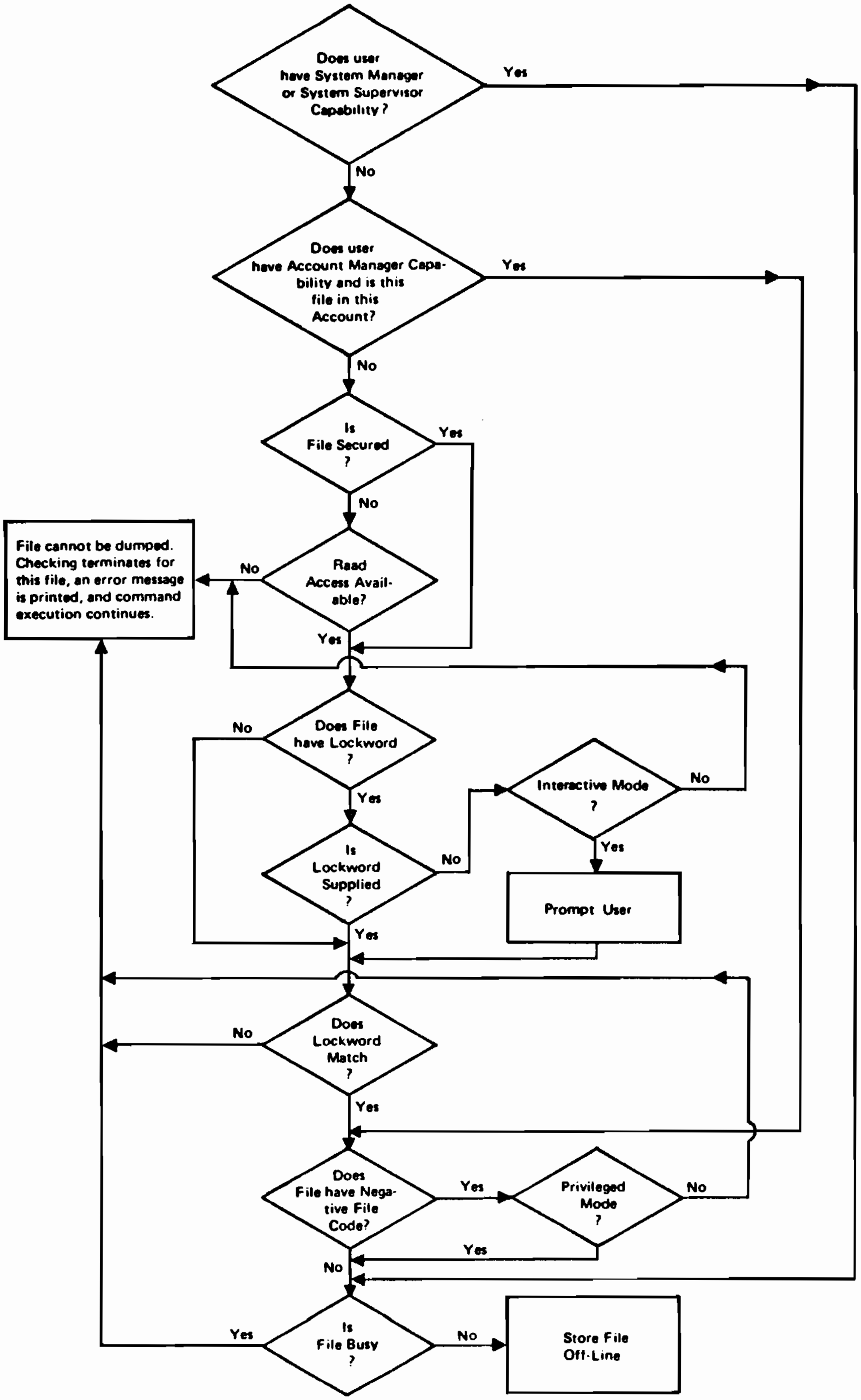
The *fileset* parameter allows the user to request dumping of one file alone, or various sets of files. It contains three positional-fields separated by periods: the file, group, and account fields. The file field permits the user to indicate a specific file or all files within the units designated by the other fields. The group field denotes the group to which the files belong. This can be the user's log-on group, any other group, or all groups within the accounts specified by the account field. The account field indicates the account or accounts to which the groups belong. This can be the log-on account, any other account, or all accounts in the system. (To specify *all* files, groups, or accounts, the user enters the character @ in the appropriate field. The omission of an entry in the group or account field denotes the log-on group or account.) For the *fileset* parameter, the combinations of entries shown below are possible:

File Field	Group Field	Account Field	Entry Example	Meaning
filename	*groupname*	*accountname*	FILE.GROUP.ACCT	The file named, in the group and account designated.
filename	*groupname*		FILE.GROUP	The file named, in the group designated under the log-on account.
filename			FILE	The file named, under the log-on group.
@	*groupname*	*accountname*	@.GROUP.ACCT	All files in the group named, under the designated account.
@	*groupname*		@.GROUP	All files in the group named, under the log-on account.
@			@	All files in the log-on group.
@	@	*accountname*	@.@.ACCT	All files in all groups under the account named.
@	@		@.@	All files in all groups under the log-on account.
@	@	@	@.@.@	All user files in the system.

A typical user can dump any file to which he has read-access. If a file has a negative file code, however, the user must have the *Privileged Mode Optional Capability.*

A user with the System Manager or System Supervisor capability can dump any user file in the system. An account-manager user can dump any file in his account (but cannot dump those with negative file codes unless he also has the Privileged Mode Capability).

Files currently open for output, input/output, update, or append access cannot be acted upon by a :STORE command. Files currently being stored or restored cannot be acted upon by a :STORE command. However, files loaded into memory (currently running programs) and files open for input only can be stored, since their contents cannot be altered.


Does user have System Manager or System Supervisor Capability?
Yes
No
Does user have Account Manager Capability and is this file in this Account?
Yes
No
Is File Secured ?
Yes
No
File cannot be dumped. Checking terminates for this file, an error message is printed, and command execution continues.
No
Read Access Available?
Yes
No
Does File have Lockword ?
Yes
Is Lockword Supplied ?
No
Interactive Mode ?
No
Yes
Prompt User
Yes
No
Does Lockword Match ?
Yes
Does File have Negative File Code?
Yes
Privileged Mode ?
No
Yes
No
Yes
Is File Busy ?
No
Store File Off-Line

While a file is being dumped, it is locked by MPE/3000 so that it cannot be altered or deleted until safely copied to tape. If the job performing the :STORE function is aborted (through the = ABORTJOB console command), some of the files being stored may remain locked until the next cold-load.

The flow chart on the following page shows the checks performed against a file to ensure its eligibility for dumping:

After the tape is written, data showing the results of the :STORE command is printed. By default, this output is sent to the standard list device ($STDLIST). However, the user can override this default and transmit the output to another file by issuing a :FILE command equating SYSLIST (the formal file designator by which the :STORE command executor references this list file) to another file. For example, a user at a terminal might transmit this output to a line printer by entering

```
:FILE SYSLIST=MYFILE; DEV=LP
```

If the *SHOW* parameter is omitted from the :STORE command, only the total number of files actually stored, a list of files not stored, and a count of files not stored, are printed. But if *SHOW* is included, the listing of files appears, in the following format:

```
FILES STORED = xxx

     FILE          .GROUP          .ACCOUNT        LDN          ADDRESS

     filename1     .groupname1     .acctname1      ldn1         addr1
     filename2     .groupname2     .acctname2      ldn2         addr2
     .
     .
     filenamen     .groupnamen     .acctnamen      ldnn         addrn

FILES NOT STORED = yyy

     FILE          .GROUP          .ACCOUNT        FILESET      CONDITION

     filename1     .groupname1     .acctname1      fileset#     msg
     filename2     .groupname2     .acctname2      fileset#     msg
     .
     .
     filenamen     .groupnamen     .acctnamen      fileset#     msg
```

In this format, *xxx* is a value denoting the total number of files dumped onto tape; *yyy* denotes the number of files requested that were not dumped. The notations *filename*, *groupname*, and *acctname* under the FILES STORED heading name the individual files dumped, and their groups and accounts, respectively. The notation *ldn* indicates the logical device number (in decimal) of the device on which the file resides, and *addr* is the absolute address (in octal) of the file label. The notations *filename*, *groupname*, and *acctname* under the FILES NOT STORED heading, indicate the individual files not dumped, and their groups and accounts.

The notation *fileset#* shows the number of the fileset to which the particular file belongs (relative to its position in the *filesetlist* parameter). The notation *msg* is a message denoting the reason that the file was not dumped, as follows. (These errors do not abort the file storing operation, which continues.)

Message	Meaning
ACCOUNT NOT IN DIRECTORY	Specified account does not exist.
GROUP NOT IN DIRECTORY	Specified group does not exist.
FILE NOT IN DIRECTORY	Specified file does not exist.
BUSY	File is open for output, or is currently being stored or restored.
FILE CODE <0 AND NO PRIVMODE	A user without Privileged Mode Capability is attempting to STORE a file with a negative file code.
LOCKWORD WRONG	The file lockword either was not provided or was specified incorrectly.
READ ACCESS FAILURE	The user does not have read access to the specified file.

The following catastrophic errors abort the :STORE command:

Command syntax error.

Disc input/output error (in system).

File directory error.

File system error on the tape file (TAPE), list file (LIST), or temporary disc files (GOOD, ERROR) used by the :STORE command executor.

The following example illustrates the use of the :STORE command.

EXAMPLE:

To copy all files in the group GR4X (in the user's log-on account) to a tape file named BACKUP, the user enters the following commands. A listing of the files copied appears on the standard list device.

```
:FILE BACKUP; DEV=TAPE
:STORE @.GP4X; *BACKUP; SHOW
```

Explicit or implicit redundancies are permitted in *filesetlist*, but once a file has been locked down, any subsequent reference to it in *filesetlist* results in the message BUSY even though execution of the :STORE command continues.

EXAMPLE:

Suppose the file identified as FN.GN.AN is a member of the fileset referenced by @ in the following command.

```
:STORE @, FN.GN.AN; *DUMPTAPE; SHOW
```

The command is executed successfully, but the fileset# and msg notations under FILES NOT STORED on the listing will show:

filename	*groupname*	*acctname*	*fileset#*	*msg*
FN	*GN*	*AN*	*2*	*BUSY*

This same file will also be noted under FILES STORED.

RETRIEVING DUMPED FILES

The user can read back into the system, onto disc, any file, fileset, or filesetlist that has been stored off-line (on tape) by :STORE. The files referenced are attached to the appropriate groups and accounts, with previous account and group names, and lockwords all re-instated. File retrieval is requested with the :RESTORE command. This command does not create any new accounts or groups. Any tape file to be restored will only be restored if the account name and group name exist on disc (in the system directory).

:RESTORE tapefile [;[filesetlist] [;KEEP] [;DEV=device] [;SHOW] . . .]

tapefile The name of the tape file on which the *filesetlist* to be retrieved now resides. This file must be referenced in the **formaldesignator* format. A message is output to the console operator telling him which tape to mount, and the device it should be mounted on. (Required parameter.)

filesetlist The file, fileset, or filesetlist to be restored from tape. Each file is restored as a permanent file in the system file domain. The parameter is written in the same format, and subject to the same constraints as the *filesetlist* parameter of the :STORE command. The number of filesets specified is limited as follows: up to 10 by account name; up to 15 by account name and group name; up to 20 by account name, group name, and file name. If the *filesetlist* parameter is omitted, the default value is @.@.@. (all files on the tape). (Optional parameter.)

KEEP A specification that if a file referenced in *filesetlist* currently exists on disc, the file on disc is kept and the corresponding tape file is not copied into the system. If KEEP is omitted, and an identically-named file exists in the system, that file is replaced. If KEEP is omitted, AND a file on tape is eligible for restoring AND a file of the same name exists on disc AND this disc file is busy, the disc file is kept and the tape file is not restored. (Optional parameter.)

device The device class name or logical device number of the device on which all files are to be restored. If omitted, an attempt is made to replace the files on the same device type (and subtype) on which they were originally stored in the system; if this cannot be done, an attempt is made to restore the files to the same device type (ignoring sub-type); if this fails, an attempt is made to restore them to the device class DISC; if this fails, the files are not restored. (Optional parameter.)

SHOW A request that the names of the restored files be listed. If *SHOW* is omitted, only the total number of files restored, a list of the files not restored, and a count of the files not restored, are listed. (Optional parameter.)

A user with System Manager or System Supervisor Capability can restore any file from a :STORE tape, assuming the account and group to which the file belongs, and the user who created the file, exist in the system. A user with account-manager capability can restore any file in his account (but cannot restore those with negative file codes unless he also has the privileged mode capability). Any other user can restore any tape file in his log-on account if he has save-access to the group to which the file belongs (but cannot restore those with negative file codes unless he also has the privileged mode capability). If the file on tape is protected by a lockword, the lockword must be supplied in the :RESTORE command. (Users logged-on interactively are prompted for omitted lockwords.)

If a copy of a file to be restored already exists on disc, the user must have write access to the disc file (since it will be purged by :RESTORE). If this disc copy has a negative file code, the user must have privileged mode capability to restore it.

Files *currently* open, loaded into memory, or being stored or restored, cannot be acted upon by a :RESTORE command.

The :RESTORE command performs the same checking performed by the :STORE command, to ensure a file's eligibility for retrieval. If the SHOW parameter is included in the :RESTORE command, a listing is produced showing which files were restored. Otherwise, a count of files restored, a list of files not restored, and a count of files not restored, are supplied.

As with the listing produced by :STORE, the listing output by :RESTORE is transmitted to a file whose formal designator is SYSLIST; if the user does not specify otherwise, this file is equated, by default, to the standard list device ($STDLIST). The listing appears in the format shown below.

```
    FILES RESTORED = xxx

         FILE          .GROUP          .ACCOUNT        LDN        ADDRESS

         filename1     .groupname1     .acctname1      ldn1       addr1
         filename2     .groupname2     .acctname2      ldn2       addr2
         .
         .
         .
         filenamen     .groupnamen     .acctnamen      ldnn       addrn

   FILES NOT RESTORED = yyy

         FILE          .GROUP          .ACCOUNT        FILESET    CONDITION

         filename1     .groupname1     .acctname1      fileset#   msg
         filename2     .groupname2     .acctname2      fileset#   msg
         .
         .
         .
         filenamen     .groupnamen     .acctnamen      fileset#   msg
```

In this format, *xxx* denotes the total number of files restored; *yyy* denotes the number of files requested that were not restored. The notations *filename*, *groupname*, and *acctname* under the FILES RESTORED heading name the individual files restored, and their groups and accounts, respectively. The notation *ldn* indicates the logical device number (in decimal) of the device on which the file now resides, and *addr* is the absolute address (in octal) of the file label. The notations *filename*, *groupname*, and *acctname* under the FILES NOT RESTORED heading, indicate the individual files not restored, and their groups and accounts. The notation *fileset#* shows the number of the fileset to which the particular file belongs (relative to its position in the *filesetlist* parameter.) The notation *msg* is an error message denoting the reason that the file was not restored, as follows. (These errors do not abort the file-restoring operation.)

Message	Meaning
ACCOUNT DIFFERENT FROM LOGON	The file's account name is different from the name of the user's log-on account. (Users do not have save-access to groups outside of their log-on accounts.)
ACCOUNT DISC SPACE EXCEEDED	The account's disc space limit would be exceeded by restoring this file.

Message	Meaning
ACCOUNT NOT IN DIRECTORY	The account specified does not exist in the system.
ALREADY EXISTS	A copy of the file specified already exists on disc, and KEEP was also specified. The file was not replaced.
BUSY	The disc file is open, loaded, or being stored or restored at present.
CATASTROPHIC ERROR	A catastrophic error occurred while the system was restoring either this file or one previous to it on the tape, and the :RESTORE command was aborted. (Examples of such catastrophic errors are listed below.)
CREATOR NOT IN DIRECTORY	The creator of the file is not defined in the system.
DISC FILE CODE <0 AND NO PRIV MODE	One of the files (on disc) to be replaced has a negative file code, and the user does not have Privileged Mode Capability.
DISC FILE LOCKWORD WRONG	The disc file has a lockword that does not match the lockword for the file on tape.
GROUP DISC SPACE EXCEEDED	The group's disc space limit would be exceeded by restoring this file.
GROUP NOT IN DIRECTORY	The group specified does not exist in the system.
NOT ON TAPE	The file specified is not on the tape.
OUT OF DISC SPACE	There is insufficient disc space to restore this file.
SAVE ACCESS FAILURE	The user does not have save-access to the group to which the file belongs.
TAPE FILE CODE <0 AND NO PRIV MODE	One of the files (on tape) to be restored has a negative file code, and the user does not have Privileged Mode Capability.
TAPE FILE LOCKWORD WRONG	The tape file has a lockword that was not supplied by the user, or was specified incorrectly.
WRITE ACCESS FAILURE	The user does not have write-access to the copy of the file on disc.

The following catastrophic errors abort the :RESTORE command:

Command syntax error.

Disc input/output error (in system).

File directory error.

File system error on the tape file (TAPE), list file (LIST), or any of the three temporary files (GOOD, ERROR, and CANDIDAT) used by the :RESTORE command executor.

Improper tape; the tape used for input was not written in :STORE/:RESTORE format.

No continuation reel; the computer operator could not find a continuation reel for a multi-reel tape set.

Device reference error; the specification for the *device* parameter is illegal, or the device requested is not available.

Too many *filesets* specified.

EXAMPLE:

To retrieve from the file named BACKUP all files belonging to the user's log-on group, the user enters:

```
:FILE BACKUP; DEV=TAPE
:RESTORE *BACKUP; @; KEEP; DEV=MDISC; SHOW
```

If a tape file satisfying the @ specification already exists in the system, it is not restored.

NOTE: Tapes created by the :STORE command and :SYSDUMP command are compatible. Thus, a tape dumped through :SYSDUMP can be used as input for the :RESTORE command.

APPENDIX I
File Security

Associated with each account, group, and individual file, there is a set of security provisions that specifies any restrictions on access to the files in that account or group, or to that particular file. (Notice that these provisions apply to disc files only.) These restrictions are based upon two factors:

1. Modes of Access (reading, writing, or saving, for example).

2. Types of Users (users with Account Librarian or Group Librarian Capability, or creating users, for instance) to whom the access modes specified are permitted.

The security provisions for any file describe *what modes of access* are permitted to *which users* of that file.

The access modes possible, the mnemonic codes used to reference them in MPE/3000 commands relating to file security, and the complete meanings of these modes are listed below:

Access Mode	Mnemonic Code	Meaning
Reading	R	Allows users to read files.
Locking	L	Permits a user to prevent concurrent access to a file by himself and another user. Specifically, it permits use of the FLOCK and FUNLOCK intrinsics, and the exclusive-access option of the FOPEN intrinsic, all described in *HP 3000 Multiprogramming Executive Operating System (03000-90005).*
Appending	A	Allows users to add information and disc extents to files, but prohibits them from altering or deleting information already written. This access mode implicitly allows the locking (L) access mode described above.

Access Mode	Mnemonic Code	Meaning
Writing	W	Allows users general writing access, permitting them to add to, delete, or change any information on files. This includes removing entire files from the system with the :PURGE command. Writing access also implicitly allows the locking (L) and appending (A) access modes described previously.
Saving	S	Allows users to declare files *within a group* permanent, and to rename such files. This ability includes the creation of a new permanent file with the :BUILD command.
Executing	X	Allows users to run programs stored on files, with the :RUN command or CREATE intrinsic.

The types of users recognized by the MPE/3000 security system, the mnemonic codes used to reference them, and their complete definitions are listed below.

User Type	Mnemonic Code	Meaning
Any User	ANY	Any user defined in the system; this includes all categories defined below.
Account Librarian User	AL	User with Account Librarian Capability, who can manage certain files within his account that may or may not all belong to one group.
Group Librarian User	GL	User with Group Librarian Capability, who can manage certain files within his home group.
Creating User	CR	The user who created this file.
Group User	GU	Any user allowed to access this group as his log-on or home group, including all GL users applicable to this group.
Account Member	AC	Any user authorized access to the system under this account; this includes all AL, GU, GL, and CR users under this account.

Users with System or Account Manager Capability bypass the standard security mechanism; a System Manager User always has R, A, W, L, X access to any file in the system, and S access to any group in his account; an Account Manager User always has unlimited (R, A, W, L, X) access to any file in his account and S access to any group in his account.

The user-type categories that a user satisfies depend on the file he is trying to access. For example, a user accessing a file that is not in his home group is not considered a group librarian for this access even if he has the Group Librarian User Attribute.

Notice that in order to extend a file, either W or A access to that file is required.

> *NOTE:* *In addition to the above restrictions, in force at the account, group, and file level, a file lockword can be specified for each file. Users must then specify the lockword as part of the filename to access this file. The way in which lockwords are assigned to files is discussed in HP 3000 Multiprogramming Executive Operating System (03000-90005).*

The security provisions for the account and group levels are managed only by users with the System Manager and the Account Manager Capabilities respectively, and can only be changed by those individuals. The manner in which they are implemented is described in Section VI. The provisions themselves are summarized in this section.

Account-Level Security

The security provisions that broadly apply to all files within an account are set by a System Manager User when he creates the account. The initial provisions can be changed at any time, but only by that user.

At the account level, five access modes are recognized:

Reading (R)

Appending (A)

Writing (W)

Locking (L)

Executing (X)

Also, at the account level, two user types are recognized:

Any User (ANY)

Account Member (AC)

If no security provisions are explicitly specified for the account, the following provisions are assigned by default:

- For the system account (named SYS), through which the System Manager User initially accesses the system, reading and executing access are permitted to all users; appending, writing, and locking access are limited to account members. (Symbolically, these provisions are expressed as follows:

  (R,X:ANY; A,W,L:AC)

  In this format, colons are interpreted to mean ": . . . is permitted only to . . .", or ". . . is limited to . . .". Commas are used to separate access modes or user types from each other. Semicolons are used to separate entire access mode/user type groups from each other.)

- For all other accounts, the reading, appending, writing, locking, and executing access are limited to account members. (R,A,W,L,X: AC).

Group-Level Security

The security provisions that apply to all files within a group are initially set by an Account Manager User when he creates the group. They can be equal to or more restrictive than the provisions specified at the account level. (The group's security provisions also can be less restrictive than those of the account — but this effectively results in *equating* the group restrictions with the account restrictions, since a user failing security checking at the account level is denied access at that point, and is not checked at the group level.) The initial group provisions can be changed at any time, but only by an account-managing user for that group's account.

At the group level, six access modes are recognized:

Reading (R)

Appending (A)

Writing (W)

Locking (L)

Executing (X)

Saving (S)

Also, at the group level, five user types are recognized:

Any User (ANY)

Account Librarian User (AL)

Group Librarian User (GL)

Group User (GU)

Account Member (AC)

If no security provisions are explicitly specified, the following provisions apply by default.

- For a public group (named PUB), whose files are normally accessible in some way to all users within the account, reading and executing access are permitted to all users; appending, writing, saving, and locking access are limited to account librarian users and group users (including group librarian users). (R,X:ANY; A,W,L,S:AL,GU).

- For all other groups in the account, reading, appending, writing, saving, locking, and executing access are limited to group users. (R,A,W,L,X,S:GU).

File-Level Security

When a file is created, the security provisions that apply to it are the default provisions assigned by MPE/3000 at the file level, coupled with the user-specified or default provisions assigned to the account and group to which the file belongs. At any time, however, the creator of the file (and *only* this individual) can change the file-level security provisions, as described in *HP 3000 Multiprogramming Executive Operating System (03000-90005).* Thus, the total security provisions for any file depend upon specifications made at all three levels — the account, group, and file levels. A user must pass tests at all three levels — account, group, and file security, in that order — to successfully access a file in the requested mode.

If no security provisions are explicitly specified by the user, the following provisions are assigned at the file level by default:

- For all files, reading, appending, writing, locking, and executing access are permitted to all users. (R,A,W,L,X:ANY).

Because the total security for a file always depends on security at all three levels, a file not explicitly protected from a certain access mode at the file level may benefit from the default protection at the group level. For example, the default provisions at the file level allow the file to be read by any user — but the default provisions at the group level allow access only to group users. Thus, the file can only be read by a group user.

In summary, the default security provisions at the account, group, and file levels combine to result in these *overall* default security provisions:

Filereference	File	Access Permitted	Save Access to Group
filename.PUB.SYS	Any file in Public Group of System Account.	(R,X:ANY; W:AL,GU)	AL,GU
filename.group-name.SYS	Any file in any group in System Account.	(R,W,X:GU)	GU
filename.PUB.ac-countname	Any file in Public Group of any account.	(R,X:AC; W:AL,GU)	AL,GU
filename.group-name.accountname	Any file in any group in any account.	(R,W,X:GU)	GU

Stated another way, when the default security provisions are in force at all levels, the standard user (without any other user attributes) has:

- Unlimited access (in all modes) to all files in his log-on group and home group.
- Reading and executing access (only) to all files in the public group of his account and the public group of the System Account.

The user cannot access any other file in the system (in any mode).

A user can only create files within his own account.

APPENDIX J
Account/Group/User Entries

The format of the standard entry for each account is shown in Figure J-1. In this format,

- ANAME and APASS are eight-character names, right-padded with blanks.
- The double-word numeric quantities are double-word integers, with %17777777777 representing "unlimited."
- ACAP can be decoded by reference to the WHO intrinsic, described in *HP 3000 Multiprogramming Executive Operating System (03000-90005).*
- MAX JOB PRIORITY is a numerical quantity.

The format of the entry for each group is shown in Figure J-2.

- GNAME and GPASS are eight-character names, right-padded with blanks.
- The double-word numeric quantities are double-word integers, with %17777777777 representing "unlimited."
- GCAPABILITY is in the same format as capability-class attributes (Word 2 of CAP), as returned by the WHO intrinsic.

The format of the entry defining each user is shown in Figure J-3.

- UNAME, UPASS and UHGROUP are eight-character names, right-padded with blanks.
- UCAP can be decoded by reference to the WHO intrinsic.
- JOBPRI is a numerical quantity.

ACCOUNT ENTRY (29 WORDS)

0	ANAME	0	
1		1	ACCT. NAME
2		2	
3		3	
4	AGIPNTR	4	ACCT. GROUP INDEX POINTER
5	AUIPNTR	5	ACCT. USER INDEX POINTER
6	ACAP	6	CAPABILITY
7		7	
10	ALATTR	8	LOCAL ATTRIBUTES
11		9	
12	APASS	10	PASSWORD
13		11	
14		12	
15		13	
16	ADFSCOUNT	14	DISC FILE SPACE COUNT (SECTORS)
17		15	
20	ADFSLIMIT	16	DISC FILE SPACE LIMIT (SECTORS)
21		17	
22	ACPUCOUNT	18	CPU TIME COUNT (SECONDS)
23		19	
24	ACPULIMIT	20	CPU TIME LIMIT (SECONDS)
25		21	
26	ACONTIMECOUNT	22	CONNECT TIME COUNT (MINUTES)
27		23	
30	ACONTIMELIMIT	24	CONNECT TIME LIMIT (MINUTES)
31		25	
32		26	FLAGS (SEE BELOW)
33		27	MAX. JOB PRIORITY
34		28	

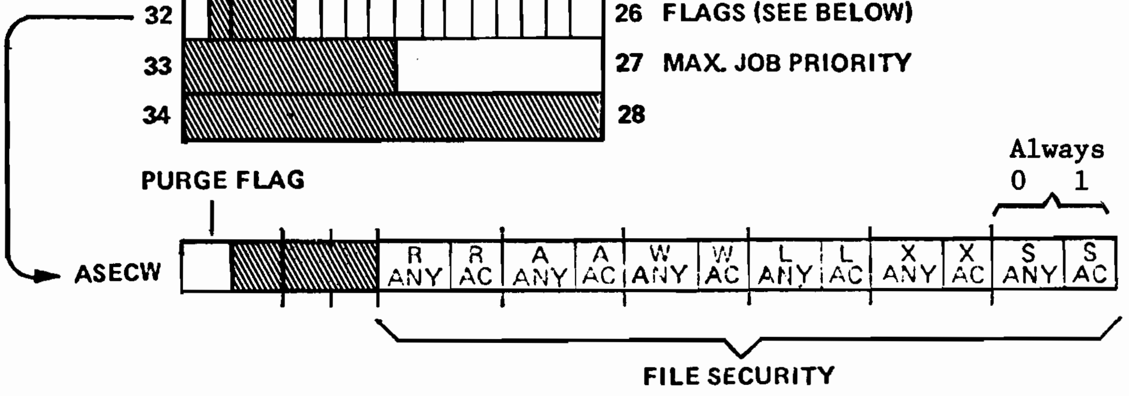
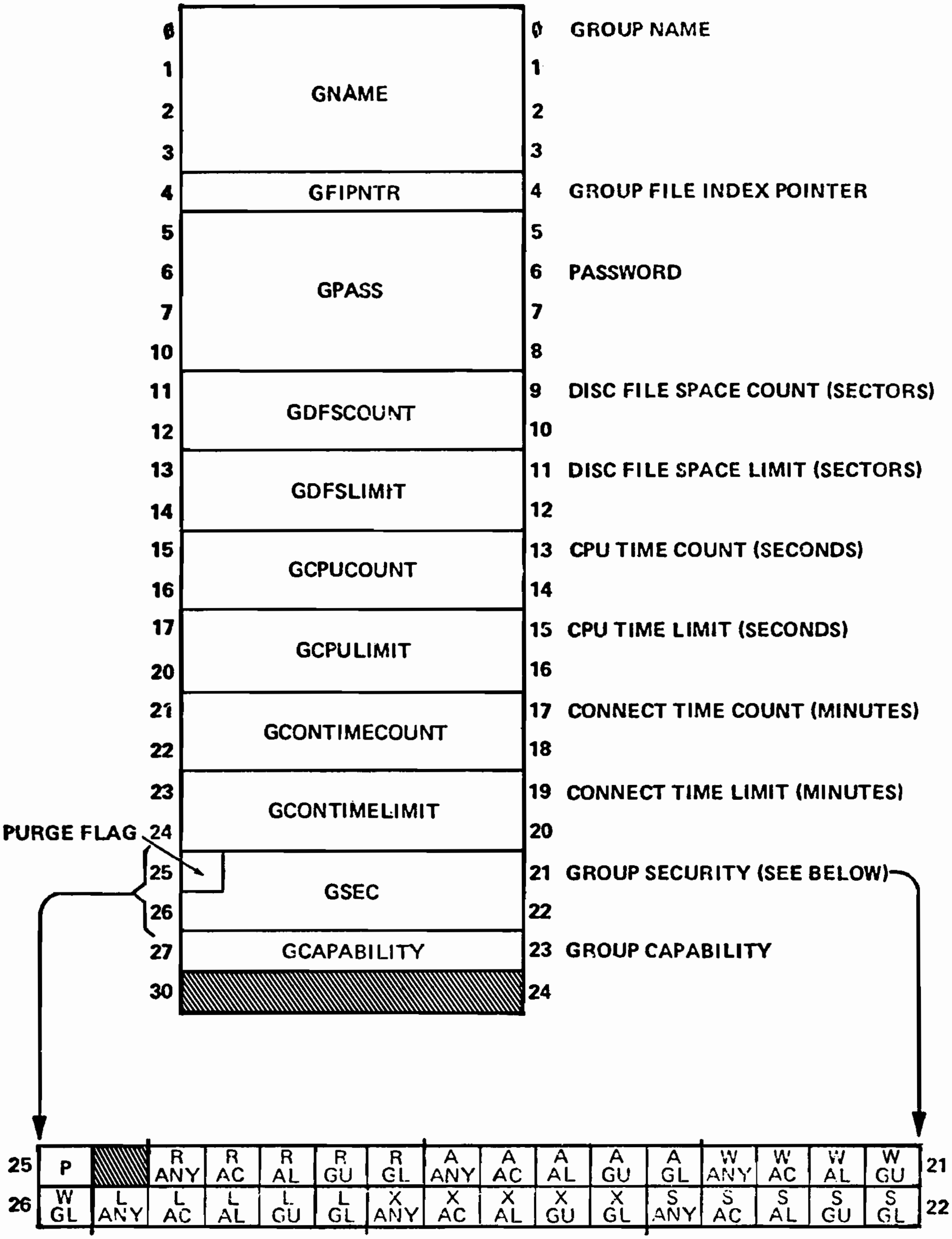


Figure J-1. Account Entry Format

GROUP ENTRY (25 WORDS)



0	GNAME	0	GROUP NAME
1		1	
2		2	
3		3	
4	GFIPNTR	4	GROUP FILE INDEX POINTER
5	GPASS	5	
6		6	PASSWORD
7		7	
10		8	
11	GDFSCOUNT	9	DISC FILE SPACE COUNT (SECTORS)
12		10	
13	GDFSLIMIT	11	DISC FILE SPACE LIMIT (SECTORS)
14		12	
15	GCPUCOUNT	13	CPU TIME COUNT (SECONDS)
16		14	
17	GCPULIMIT	15	CPU TIME LIMIT (SECONDS)
20		16	
21	GCONTIMECOUNT	17	CONNECT TIME COUNT (MINUTES)
22		18	
23	GCONTIMELIMIT	19	CONNECT TIME LIMIT (MINUTES)
24		20	
25	GSEC	21	GROUP SECURITY (SEE BELOW)
26		22	
27	GCAPABILITY	23	GROUP CAPABILITY
30	▨	24	

PURGE FLAG

25	P	▨	R ANY	R AC	R AL	R GU	R GL	A ANY	A AC	A AL	A GU	A GL	W ANY	W AC	W AL	W GU	21
26	W GL	L ANY	L AC	L AL	L GU	L GL	X ANY	X AC	X AL	X GU	X GL	S ANY	S AC	S AL	S GU	S GL	22

Figure J-2. Group Entry Format

USER ENTRY (19 WORDS)

	Ø	UNAME	Ø	USER NAME
	1		1	
	2		2	
	3		3	
	4	UCAP	4	CAPABILITY
	5		5	
	6	ULATTR	6	LOCAL ATTRIBUTES
	7		7	
	10	UPASS	8	PASSWORD
	11		9	
	12		10	
	13		11	
	14	UHGROUP	12	HOME GROUP (MAY BE NULL)
	15		13	
	16		14	
	17		15	
	20	ULOGCOUNT	16	LOGON COUNT (# OF USERS LOGGED ON)
UMAXJOBW	21	▨ JOBPRI	17	MAX. JOB PRIORITY
	22	NOT USED	18	

PURGE FLAG

Figure J-3. User Entry Format

APPENDIX K
Disc Volume Labels

When each disc is initialized, MPE/3000 writes a volume label in the first sector (Sector 0). This volume label is written in the following format:

Words	Contents
0–5	On the system disc, this field contains the bootstrap input/output program. On other discs, this field is filled with zeros.
6 (Bits 6:6)	Disc type.
(Bits 12:4)	Disc sub-type.
7	Cold-load count (incremented each time the system is cold-loaded).
8–9	The characters "3000," used to verify that the disc label is valid.
10–13	Volume name (Left-justified and padded with blanks).
14–127	Reserved (Initialized to zeros).

APPENDIX L
Defective Disc Track Table

The Defective Track Table, used by MPE/3000 to maintain a record of any defective areas on a disc, resides in the second sector (Sector 1) of each disc. It is written in the following format:

Word	Contents
0	Number of entries (n) in the table (ranges from 0 to 120).
1–n	
(Bits 0:14)	Track number.
(Bits 14:2)	Status, where 0 = Suspect track. 1 = Suspect alternate track. 2 = Deleted track. 3 = Reassigned track.
121-125	Reserved (filled with zeros).
126	Next available alternate track (moving-head discs only).
127	Logical disc pack size (in cylinders for moving-head discs, and tracks for fixed head discs).

APPENDIX M
System Disc Space Utilization

The system disc is used primarily for four system requirements: MPE/3000 virtual memory, the disc directory, MPE/3000 files, and information needed during cold loading. Any space remaining is available for user files.

Sectors 0 through 19 are reserved for the disc volume label, defective tracks table, and bootstrap input/output program. The disc free space table starts in Sector 20 and ranges in length between 4 and 16 sectors, depending on the type of disc.

Following the free space table are the disc directory and virtual memory. Their sizes are defined by parameters specified at system configuration time. (See below for the guidelines for determining the sizes of these two areas.)

The next portion of the disc is used for the system files, message catalog, and tables used by INITIAL and INITIAL program segments. At the present time, this accounts for approximately 2350 sectors. (This includes the SL file containing the Scientific, Compiler and COBOL/3000 Libraries.)

The disc directory currently has the following approximate maxima: 650 accounts; 155 groups per account; 200 users per account; and 1385 files per group. The following formula should be used to determine the approximate number of sectors needed for the directory:

$$\text{Sectors} = 6+3\left\lceil\frac{A}{10.15}\right\rceil+2A\left(1+\left\lceil\frac{U}{10.15}\right\rceil+\left\lceil\frac{G}{7.8}\right\rceil+G\left(1+\left\lceil\frac{F}{32.5}\right\rceil\right)\right)$$

where A = Number of accounts in the system

G = Average number of groups per account

U = Average number of users per account

F = Average number of files per group

Virtual memory is used only for the swapping of data segments. It is allocated in 4-sector blocks. Approximately 250 sectors are needed for system data segments. For each user on the system, the following amount of virtual memory is needed:

32 sectors for Command Interpreter stack

8 sectors for all unbuffered files

4 sectors per open buffered file

16 sectors for system area in the user's stack plus 4 sectors per 512 words in the DL/Z area of the stack

In addition, while a program is being loaded, 40 sectors of virtual memory are needed for the Loader.

To determine the amount of virtual memory needed, estimate the average number of concurrent users, average stack size, average number of buffered files open per user, and number of users who will be loading programs at the same time; then use these figures in conjunction with the above values.

APPENDIX N

:STORE Tape Format

The format of a tape created by the :STORE command is:

Item No.	Item
1	End-of-File (EOF) Mark
2	EOF Mark
3	Header label (40 words), used as follows:

Words	Contents	
0–13	"STORE/RESTORE LABEL-HP/3000."	
14–22	Unused by MPE/3000.	
23	Reel number.	
24	Bits (0:7) = last 2 digits of year (7:9) = Julian date	Date of creation.
25	Bits (0:8) = hours (8:8) = minutes	Time of creation.
26	Bits (0:8) = seconds (8:8) = tenth-of-seconds	

Item No.	Item
4	EOF Mark
5	Tape directory—Consists of 12-word entries, blocked 85 per 1020—word record. (The last record may be shorter.) There is one entry for each file on the tape, and the entries are ordered the same as the files. The 12-word entry is:

Word	Contents
0–3	File name
4–7	Group name
8–11	Account name

Item No.	Item
6	EOF Mark

Item No.	Item
7	First file. The data is blocked with 1024 words per physical tape record. (The last record may be shorter, but will always be a multiple of 128 words.) For fixed-length and undefined-length record files, only data up to the end-of-file is dumped; intervening zero-length extents are not dumped. For variable-length record files, only allocated extents are dumped.
8	EOF Mark
9	Second File
10	EOF Mark
	· · ·
11	Last File
12	EOF Mark
13	Trailer Label (40 words). Identical to header label (Item 3) except that Words 21 and 22 are used as follows:

Word	Use
21	=1 means that preceding file ended with preceding EOF mark
22	=1 means that entire tape set ends with preceding EOF mark

Item No.	Item
14	EOF Mark
15	EOF Mark
16	EOF Mark

:STORE tapes may have multiple reels. If end-of tape (EOT) is detected during a write data operation, a file mark is written followed by Items 13 to 16 above, with word 21 of the trailer label set to 1 if this was the last record of the file and 0 otherwise. If EOT is detected on a write file mark operation, Items 13 to 16 are written, with word 21 set to 1 and word 22 set to 1 if this is the last file on the tape, and 0 otherwise. Reels subsequent to Reel 1 have the following format:

1. Header label
2. EOF mark
3. Remainder of preceding file or next file
4. EOF mark
5. Next file; the rest of the tape is written in the same format as the first reel.

APPENDIX O
Cold-Load Tape Format

Cold-load tapes, created by the :SYSDUMP command (discussed in Section IV), can be entered into the system by the :RESTORE command as well as by the cold-loading operation. These tapes have the following format:

Item No.	Item
1	Bootstrap—A record, 40 words long, read by the cold-load microcode and containing a six-word SIO program which reads in the next record.
2	I/O program—An input/output program to read the records defined in Items 3 through 18.
3	ICS and entry point (34 words)—First 32 words are the Interrupt Control Stack (ICS) containing the initial values for DB, DL, Z, Q and S. Words 32-33 are the SETR and EXIT instructions which will be executed when the RUN switch is pressed.
4	Low-core (12 words). Initial values for absolute locations 0-11.
5	Disc Cold Load Information Table (128 words)—Sector 18 of the system disc.
6	Old Volume Table (280 words)—Volume table as it was before any changes made to it on this :SYSDUMP.
7	Volume Table (280 words)—Modified volume table.
8	Logical-Physical Device Table (512 words)—as updated by input/output configuration changes.
9	Logical Device Table (2 records of 640 words each)—As updated by input/output configuration changes.
10	Device Class Table (2 records of 768 words each)—As updated by input/output configuration changes.
11	Driver Table (2 records of 640 words each). Driver name for each DRT and core-resident flag, as updated by configuration changes.
12	Core Size Related Configuration Table (640 words)—128 words for each of the 5 configurable core sizes.

Item No.	Item
13	Configuration Table (128 words)—Configuration information not related to core size.
14	Internal Interrupts (64 words)—Halt instruction for each of the internal interrupts.
15	Temporary CST (128 words)—Code segment table (CST) used for INITIAL's segments while INITIAL is running.
16	INITIAL's DB area (Blocked 1024 words per record)—Initialized DB area for INITIAL.
17	INITIAL marker (12 words)—Initial stack marker for INITIAL program.
18	INITIAL Segments 16-5 (Blocked 1024 words per record for each segment)—Those segments of INITIAL which are read in by the cold-load input/output program.
19	INITIAL Segments 4-0 (Blocked 1024 words per record for each segment)—Those segments of INITIAL which are not in core when the machine halts following cold load.
20	Message catalog (Blocked 1024 words per record)—Table of MPE/3000 messages.
21	If any files are to be dumped by :SYSDUMP, this is the current RIN table, blocked 1024 words per record and the used portion of the disc directory, blocked 1024 words per record.
22	End-of-File mark.
23	System files—Each allocated extent of each of the system files is dumped, blocked 1024 words per record. Files always dumped if they are on the disc (all in PUB.SYS) are: SL, SYSDUMP, INITIAL, CONFDATA, SEGPROC, SEGDVR, DISPATCH, LOAD, MAPP, UCOP, PROGEN, DEVREC, ININ, LOG, EXIN, IOTERM0, IOCLTTY0, IOCDRD0, IOLPRT0, IOTAPE0, IOFDISK0, IOMDISK0. In addition, any input/output drivers defined in the configuration but not in the above list are dumped.
24	End-of-File Mark.
25	If any user files were to be dumped, the rest of the tape is identical to the :STORE tape format described in Appendix N, (beginning with Item 3, the header label). The presence of the two file marks written as Items 22 and 24 above, results in :STORE and :SYSDUMP tape compatibility for use by :RESTORE. If no user files were to be dumped, the remainder of the tape is the same as a :STORE tape with no files, consisting of a Header label, two end-of-file marks, a trailer label, and three end-of-file marks.

Index

Index of Commands and Intrinsics