HP 3000 Computer Systems

HEWLETT PACKARD

A family of compatible business systems for distributed data processing

General Information Manual

Effective June 1, 1980

HP 3000 Series 30

HP 3000 Series 33

HP 3000 Series III

HP 3000 Computer Systems
General Information Manual

19447 Pruneridge Avenue
Cupertino, California 95014

Notice

The information contained in this document is subject to change without notice.

HEWLETT-PACKARD MAKES NO WARRANTY OF ANY KIND WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. Hewlett-Packard shall not be liable for errors contained herein or for incidental or consequential damages in connection with the furnishing, performance or use of this material.

Hewlett-Packard assumes no responsibility for the use or reliability of its software on equipment that is not furnished by Hewlett-Packard.

Preface

Today's business decisions cannot be based on yesterday's data. Immediate access to the most up-to-date information is a necessity for the financial planning, sales forecasting, production scheduling, and other complex tasks which are part of each business day. The HP 3000 interactive business computer systems are uniquely qualified to meet these demands for accurate, timely information. Designed specifically for terminal-oriented business data processing, the HP 3000 systems are based on an integrated hardware and software concept. HP 3000 computer systems are delivered ready to run application programs. Complete data base management and data entry and inquiry facilities are included to make the basic HP 3000 system a fully capable and operational system.

For manufacturers, Hewlett-Packard offers user-customizable materials planning and control software. Ten functionally integrated modules are available to address all of the important phases of materials planning and control. This interactive, data base oriented HP product can help you improve inventory management, control costs, and obtain more timely and accurate information on which to base purchasing and manufacturing decisions.

For in-house software development, Hewlett-Packard provides tools for efficient interactive program development. With six programming languages, debugging aids, programming utilities, and high level procedures for data base management, data entry, data inquiry and other programming tasks, programmer productivity on the HP 3000 is high.

Advanced distributed processing capabilities through a wide variety of data communications products allow HP 3000 systems to bring information directly to the people who need it, when they need it.

- This manual is a thorough discussion of the HP 3000 transaction processing environment, manufacturing application programs, data communications, operating system and system architecture. The appendices outline the various operating system and machine level commands and intrinsics, and expand on the hardware features of the computer systems.
- A separate brochure (5953-0588) contains reference sheet specifications of the hardware systems, Fundamental Operating Software, optional software subsystems and hardware peripherals, software/hardware support services and training course descriptions.

Contents

Chapter 1. System Introduction

Chapter 2. Transaction Processing

Chapter 3. Manufacturing Environments

Chapter 4. Data Communications

Chapter 5. The Operating System—MPE III

Chapter 6. System Architecture

Appendices

Chapter 1

SYSTEM INTRODUCTION

MANAGEMENT OVERVIEW
SYSTEM COMPONENTS
FUNDAMENTAL OPERATING SOFTWARE
LANGUAGES
DATA COMMUNICATIONS
MANUFACTURING APPLICATIONS
DOCUMENTATION
TRAINING
SOFTWARE SUPPORT SERVICES
HARDWARE SUPPORT SERVICES
HP GENERAL SYSTEMS USERS GROUP

MANAGEMENT OVERVIEW

HP 3000 computer systems are a compatible family of interactive, data-base oriented business processing systems. They have an integrated hardware and software design ideally suited to multi-purpose business data processing and dedicated manufacturing applications. Their advanced design provides a wide range of user-oriented capabilities that can be utilized with equal ease on both stand-alone systems and those in distributed processing networks.

HP 3000 systems can be configured in three different ways: as a ready-to-run production system, as a program development system, and as an integrated element of a distributed data processing network.

Ready-To-Run Production Systems

HP 3000 computers are delivered as complete systems with all the hardware and software necessary to run and operate applications programs immediately. In addition to hardware, every HP 3000 computer system includes Fundamental Operating Software (FOS) which makes it a completely functional production system. Included in FOS is the Multi-Programming Executive operating system (MPE) that schedules, monitors, and controls processing, and a complete set of software subsystems for data management, data entry and other capabilities needed in business processing applications. The Fundamental Operating Software enables you to immediately run application programs developed and compiled on any HP 3000 without installing any additional software on the executing system other than the program itself. One HP 3000 can satisfy the program development needs of an entire network of HP 3000 Computer Systems.

Program Development Systems

Optional software subsystems, such as language compilers, are available for program development. Highly compatible interfaces between subsystems have been created so that programmers can use the Fundamental Operating Software, in conjunction with the compilers, as a high-level program development tool. The subsystems of FOS include a wide range of high-level procedures for data management and data entry that are callable from any of the HP 3000 programming languages. This enables programmers to give application programs comprehensive capabilities and makes program development on the HP 3000 quick and convenient.

Distributed Data Processing Systems

Data communication subsystems are also available to enable you to set up distributed processing networks that conform to the way you want to do business. A variety of terminal communications products allow you to place terminals wherever you wish without regard to distance or location. HP Distributed Systems software permits sharing of resources between HP 3000 computer systems and other Hewlett-Packard computers. HP 3000 systems can also provide batch access to IBM mainframe computers. These capabilities are supplied as high level services which free the programmer from complex tasks and allow you to place the processing power where you need it—regardless of the level of expertise at the remote site.

Combined Function Systems

The above three HP 3000 configurations are by no means mutually exclusive. All HP 3000 computer systems can simultaneously perform transaction processing operations, interactive program development, batch processing and data communications (see Figure 1-1).

You can distribute your processing according to functional needs or geographical location while retaining the degree of control and security you desire. At each location, you can specify the function and capability you want your HP 3000 to have: a single dedicated application, multiple applications, program development or a combined general use system.

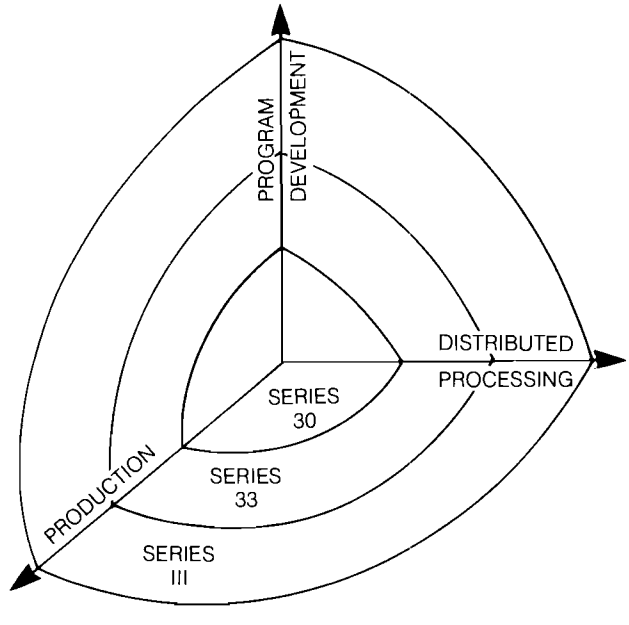
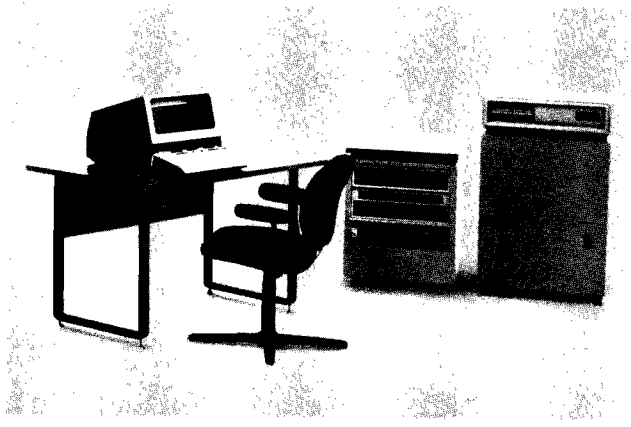


Figure 1-1. Each HP 3000 computer system can be configured to provide the exact mix of capabilities you require along three functional dimensions: Production, Distributed Data Processing, and Program Development.

HP 3000 computer systems are available in three models:



Series 30

Series 33

Series III

A Commitment to Your Success

Hewlett-Packard is committed to making our products easy to use by non-data processing professionals, whether through HP supplied application programs or the interactive data access facilities for terminal users. Also, the wide range of program development tools makes it easy for programmers to create user-oriented programs.

Superior hardware and software is not enough. Comprehensive support services are available to assist you in maintaining all phases of your operation. Among them are the Remote System Verification Program (RSVP) available on the Series 30 and 33 that permits these systems to be diagnosed remotely from an HP office by system experts. Another innovative service is the Phone-In Consulting Service (PICS) that enables your System Manager to receive immediate assistance via telephone to resolve software related problems. Personal attention from a well trained Systems Engineer is available whenever you need it. A full range of documentation, training, consulting, and support programs are offered to insure that you have the assistance you need to be successful. Hewlett-Packard supplies a full range of support services and gives you the flexibility to tailor them to your needs.

Hewlett-Packard documentation, training and support services are discussed at the end of this chapter and in the reference sheets in Section II.

SYSTEM COMPONENTS

An HP 3000 computer system consists of the system hardware, the fundamental operating software, and additional software subsystems, all of which contribute to the easy development and operation of user applications. Hewlett-Packard also provides a set of ready-to-use applications developed for a manufacturing environment. These components and their relationships are illustrated in Figure 1-2.

An Integrated System

The fundamental HP 3000 computer system consists of the system hardware, and the Fundamental Operating Software that includes the MPE operating system, utility programs, and data management, and data entry subsystems.

The HP 3000 software is integrated with an advanced hardware design that includes stack architecture, variable length code segmentation, a hardware-assisted virtual memory scheme, user protection, and dynamic storage allocation. Hardware and software work together, with hardware performing many of the operations conventionally performed by software, such as interrupts or subroutine calls.

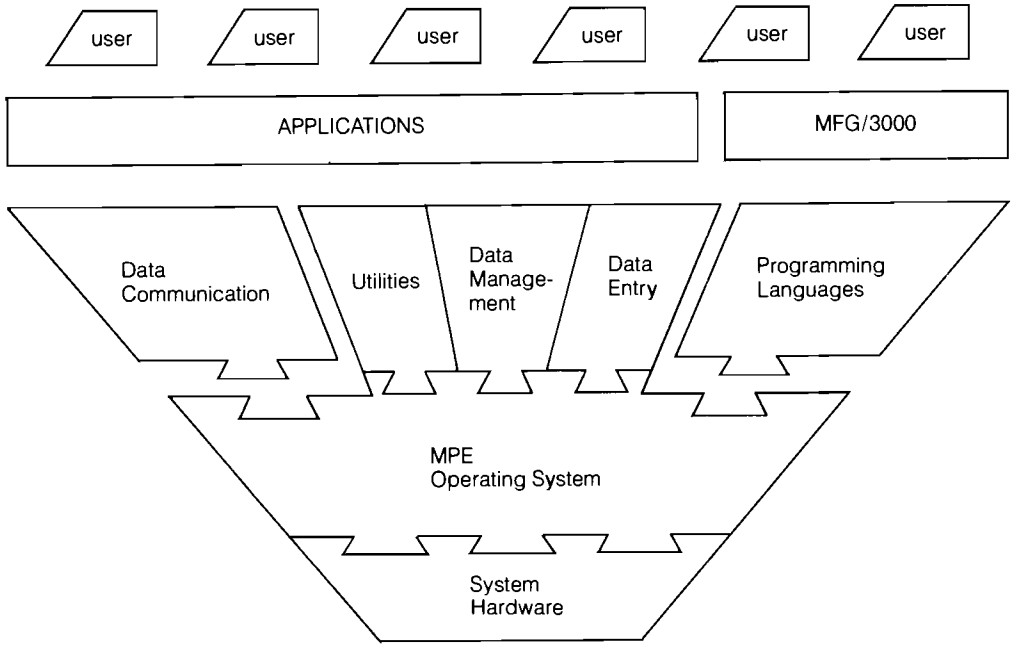


Figure 1-2. HP 3000 Computer System: System Components

System Hardware

The HP 3000 incorporates many features otherwise found only on very large computer systems, including:

- Stack Architecture—provides private, hardware protected data storage for each user plus automatic movement of this data to and from the central processor.
- Virtual Memory—consists of main memory plus an extensive storage area on magnetic disc to provide a total memory far exceeding the main memory size.
- Separation of Code and Data—strictly separate domains for code and data permits code sharing and re-entrant execution while maintaining data privacy.
- Fault Control Memory—high speed semiconductor memory modules provide automatic fault detection and single-bit correction.
- Microcode—more than 200 instructions are microcoded operations in read-only memory, plus a full set of microcoded system operations.
- Concurrent I/O and CPU operation—allows input/output to be performed concurrently with CPU operation.

Although the same software is used by all HP 3000 computer systems, specific system hardware differs for the Series 30, Series 33, and Series III. Series 30 and Series 33 are described in Appendix D, Series III in Appendix F. Generally, HP 3000 system hardware includes the central processor unit (CPU), main memory, and various peripheral devices available on each system series. Refer to Chapter 6 for a full discussion of the system architecture, and the HP 3000 Price/Configuration Guide for configuration details.

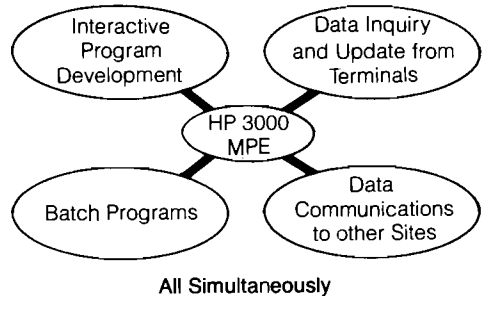


Figure 1-3. HP 3000 MPE Operating System controls and supervises all processing.

FUNDAMENTAL OPERATING SOFTWARE

MPE Operating System

The Multiprogramming Executive operating system (MPE III) is the disc-based software system which supervises the processing of all programs that run on the HP 3000. MPE dynamically allocates such system resources as main memory, the central processor, and peripheral devices to each program as needed. In addition, MPE coordinates all user interaction with the system, providing an easy to use command language interface and a powerful programmatic interface in the form of intrinsics and a versatile file system.

MPE monitors and controls program input, compilation, run preparation, loading, execution and output. It also controls the order in which programs are executed and allocates and maintains usage records of the hardware and software resources they require. By relieving you of many program control, input/output, and other housekeeping responsibilities, MPE makes the HP 3000 extremely easy to use.

The major features of the operating system are:

- Multiprogramming: concurrent transaction processing, data communications, on-line program development, and batch processing
- Virtual memory
- Concurrent multilingual capability
- Sharable code
- Device independent file system
- Input and output spooling
- Complete system security and automatic accounting of resources
- Friendly, powerful command language
- Complete interactive terminal management, both local and remote
- Automatic system restart after power failure

Details of the MPE operating system are presented in Chapter 5.

Standard in the Fundamental Operating Software of each HP 3000 is:

Data Entry

- **HP V/3000**—A comprehensive data entry and forms management subsystem that provides both a ready-to-use data entry program, ENTRY, and a programmatic interface for your terminal-oriented application programs. It includes facilities for immediate on-line entry and modification of data, a wide range of data editing and validation, record reformatting, and interactive forms design and forms management.

Data Management

The HP 3000 has extensive data management facilities with both sequential and random access methods included as part of the MPE III operating system. Additional access methods and data management capabilities included in the Fundamental Operating Software are:

- **KSAM/3000**—Keyed Sequential Access Method, an indexed file subsystem providing one primary and up to 15 alternate keys, with retrieval based upon the value of the keys.

- **IMAGE/3000**—Data Base Management System allows information to be logically related between data sets (files). It facilitates data independence, data security and integrity and minimizes data redundancy. A transaction logging facility provides an audit trail of data base activity, and a security feature permits locking at the item level. Remote data base access through HP data communication software is also included.

- **QUERY/3000**—Complements IMAGE by supplying an English-like inquiry language for entry, updating and reporting with IMAGE data bases.

Utilities

A set of utility programs, standard on each HP 3000, eases program development and file manipulation and aids in system administration. The utilities included in the Fundamental Operating Software are:

- **EDIT/3000**—A powerful and easy to use text editor
- **FCOPY/3000**—A program for general file copying
- **SORT-MERGE/3000**—A facility for ordering records in a file and merging sorted files
- **System Utilities**—provide administrative controls, reports on system resources, and other special purpose capabilities.
- Facility to execute compiled programs without the source language compiler on the system (except those written in APL\3000).

The utilities, data entry and data management software subsystems are described more fully in Chapter 2.

ADDITIONAL SOFTWARE SUBSYSTEMS

In addition to the Fundamental Operating Software, a full set of programming languages and extensive data communication facilities are offered for use on HP 3000 computer systems.

Languages

Designed to provide you with language flexibility, the HP 3000 offers six high-level programming languages which let you select the language best suited to the task. Programs can be written in:

- **COBOL**
- **RPG**
- **FORTRAN**
- **BASIC**
- **APL** (only on Series III)
- **SPL** (Systems Programming Language—a high level machine dependent language that takes full advantage of HP 3000 design features)

Applications written in different languages, or a combination of languages, can be run simultaneously. Data files and peripheral devices can be used in common by programs in any language without program changes.

Within MPE and the other subsystems of the Fundamental Operating Software reside high level procedures for terminal handling, data entry, and data management. These procedures, or intrinsics as HP refers to them, can easily be called from HP 3000 programming languages. Thus, the addition of a compiler is all that is needed to equip the HP 3000 as a program development machine. Because FOS is standard on all HP 3000's, compiled programs run on all HP 3000 computer systems and can take full advantage of the data entry, data management, and other high level program calls.

The languages are discussed in detail in Chapter 2.

Data Communications

HP 3000 Computer Systems employ a network architecture called Hewlett-Packard Distributed Systems Network (HP-DSN). HP-DSN allows diverse computer systems to be linked into information processing networks. Through the HP-DSN communications product family, data processing can be distributed among geographically or functionally dispersed computers and terminals. Flexible enough to support either a latticework of autonomous computer systems or a hierarchical configuration of satellite processors to a central computer, HP-DSN gives you the freedom to design a data processing network that uniquely fits your organizational structure. HP-DSN stresses interactive communication among processors and is implemented as a set of high level services that free the programmer from complex coding tasks and the concerns of hardware interfaces.

Data communication subsystems extend the basic asynchronous terminal communications under MPE to include:

- **MTS/3000**—synchronous multipoint terminal communications that allow users to have up to 32 terminals share a single remote or local line at speeds up to 9600 bps.

The variety of communications subsystems that link HP 3000 computers to IBM-type mainframe systems include:

- **IML/3000**—Interactive Mainframe Link/3000 (3270 Emulator) provides high-level, interactive communication between HP 3000 and IBM mainframe computers.
- **MRJE/3000**—IBM HASP/JES/ASP multileaving workstation emulation for remote job entry by multiple users simultaneously.
- **RJE/3000**—IBM 2780/3780 emulation for remote job entry.

In addition, data communications between Hewlett-Packard computer systems in an HP Distributed Systems Network is implemented through:

- **DS/3000**—Distributed Systems/3000, provides remote file and peripheral access, interactive processing and resource sharing among HP 3000 computers and other Hewlett-Packard computer systems.

These data communications subsystems are discussed in Chapter 4.

Manufacturing Application Software

Materials Management/3000 is a user customizable, interactive system for managing the materials planning and control function of a manufacturing operation. Materials Management/3000 consists of ten application software modules:

- Master Production Scheduling
- Rough Cut Resource Planning
- Parts and Bills of Material
- Routings and Workcenters
- Material Issues and Receipts
- Inventory Balance Management
- Work Order Control
- Purchase Order Tracking
- Material Requirements Planning
- Standard Product Costing

A discrete manufacturer who assembles standard, multi-piece products in lots represents the ideal candidate for Materials Management/3000. However, the software is applicable to most manufacturing operations.

Features of Materials Management/3000

- On-line data base update.
- Customizable user interface and data base.
- On-line terminal data entry, field editing, and error correction using the capabilities of HP's Data Entry and Forms Management System.
- Easy to use, on-line transaction menus.
- Automatic transaction logging.
- Tailorable data entry screens and retrievals.
- On-line assistance via "help" screens.
- Use of "intelligent" HP CRT terminals.
- Pre-defined materials data base.
- Use of proven materials management techniques.
- Advanced security capabilities.
- Automated operator functions.

An important feature of Materials Management/3000 is the ease and speed with which the user can enter, retrieve, and modify data via an interactive terminal. The user has the capability to select a variety of "menu-like" screens on the terminal to perform the tasks associated with materials planning and control.

Materials Management/3000 is available in a form which meets the input, output, and processing requirements of most manufacturing companies. Manufacturing personnel may, however, modify data entry screens, data edits, and information retrieval screens to suit their specific needs. They may also add delete, and/or modify data items in the Materials Management/3000 data base. *All of these changes can be accomplished easily and without the need for computer programming.*

Materials Management/3000 is a standard (object code) application product which is fully supported by Hewlett-Packard. A comprehensive support package is available and includes:

- 11 user reference and installation manuals.
- 2 weeks of customer training.
- Planning, implementation, and customization consulting by Hewlett-Packard Manufacturing Industry Specialists.
- A wide range of software support services.

Each of the ten Materials Management/3000 modules is related and integrated with the others. By maintaining bills of material and current information about parts, current inventory and order information, and time phased master schedule information, the customer can take advantage of material requirements planning to meet the company's production plan. Additionally, by storing labor, material, and overhead costs in the bills of material and bills of labor, the user can calculate standard costs for each item in inventory.

Users

The final element in an HP 3000 computer system is the user. You interact directly with the computer system through the software subsystems and the operating system which in turn uses the hardware and the system peripherals. You may also interact indirectly through an application program. The tasks you perform usually fall into one of the following three categories:

- On-line interaction with application programs
- Program development
- System management

All of the system resources previously described are at your fingertips when you sit down at a terminal and access an HP 3000 computer system. Hewlett-Packard is committed to making the time you spend at the terminal friendly and highly productive.

DOCUMENTATION

Included with the HP 3000 computer system is a comprehensive set of user manuals. Complete documentation is provided for the operating system, and for special tasks such as system installation and program conversion, in addition to documentation for individual subsystems.

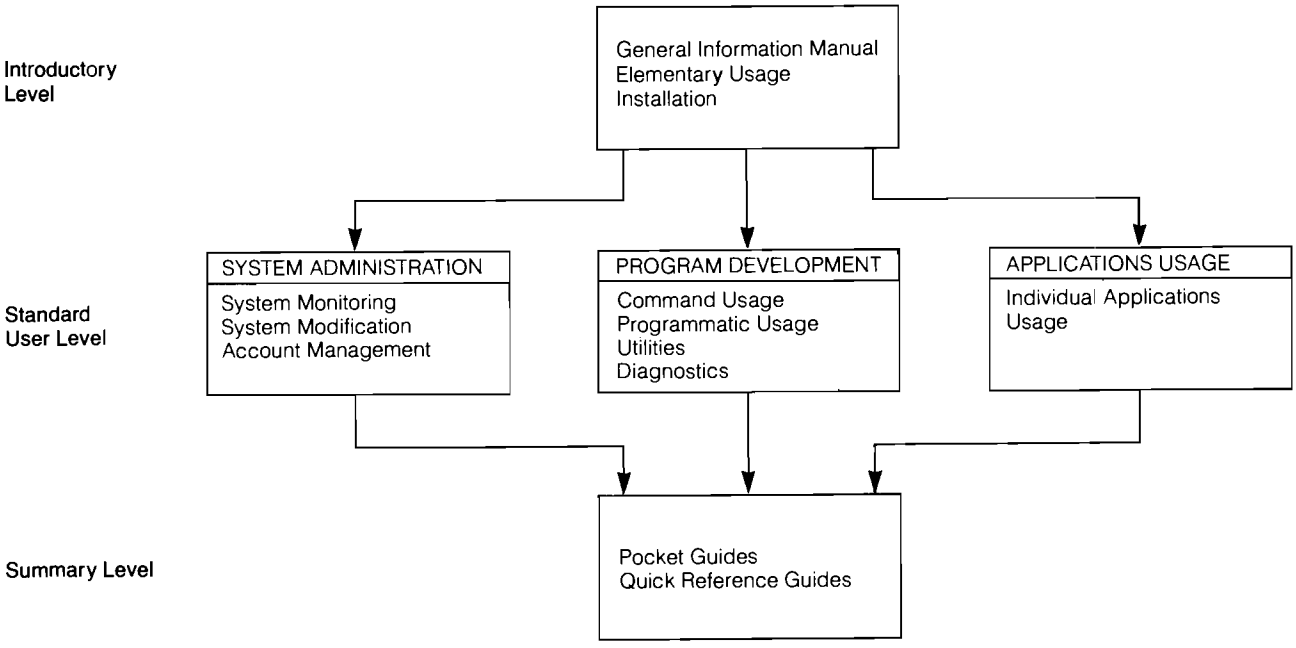


Figure 1-4. HP 3000 documentation levels

Figure 1-4 shows the three functional levels of documentation provided and how the manuals relate to one another. The full set of manuals available for the HP 3000 is presented in Appendix A in the HP 3000 Specification Guide.

Introductory Level Manuals

These manuals introduce all the major concepts of the HP 3000 computer system and provide an introduction to the use of the system. Particularly informative are the "Using" series of manuals (Using the HP 3000, Using Files, Using HP V/3000, Using COBOL) that contain practical examples showing the new user how to perform a variety of common tasks. These include how to build files, design terminal screens, compile and run programs, equate files to input/output devices, and obtain hardcopy listings.

Standard User Level Manuals

These manuals provide complete reference documentation for the full line of HP 3000 software. They fall into three general categories:

- Administrative manuals—provide a guide for day-to-day monitoring of the system by a computer operator and, on a higher level, provide the more complex information required for system management.
- Program Development manuals—provide complete reference specifications for the programmer on using the operating system, the various language compilers, the data base and file management subsystems, as well as the data editing and data entry subsystems, and all the HP 3000 utility programs. An Error Messages and Recovery Manual summarizes the error messages and recovery actions for operating system, language, and utility subsystem errors. Another useful manual, the Index to the MPE Reference Documents, provides a master index to the entire set of MPE operating system manuals.
- Applications manuals—provide full instructions for non-programmers on using the Hewlett-Packard application programs available for use on an HP 3000 computer system.

Summary Level Manuals

These are a set of convenient pocket size manuals that summarize the reference specifications for the MPE operating system, and for various programming languages and subsystems. These guides are primarily memory aids and assume a familiarity with the effects of the syntax they summarize. An essential pocket guide is the Software Pocket Guide that summarizes the MPE commands and system procedures, the file system error messages, and the commands that control the more commonly used utility programs.

A quick reference guide is available for the data entry operator who uses the HP V/3000 data entry application program, ENTRY.

TRAINING

Hewlett-Packard offers a variety of training courses that enable you to derive maximum benefit from the system's capabilities. A full curriculum of training courses spans the needs of a variety of users—from the system administrator, to the program developer, to the non-technical user of a software application.

Among the courses offered are:

- **A Programmer's Introduction**
- **System Management and Operation**
- **IMAGE/3000—Data Base Management Training**
- **KSAM/3000—Keyed Sequential Access Method**
- **HP V/3000—Screen Design and Data Entry Programming**
- **SPL/FILE System Introduction**
- **MPE III Special Capabilities**
- **DS/3000—Using Distributed Systems**
- **Materials Management/3000**

Courses may last from one to five days, and in most cases, can be presented at your facility or at an HP Technical Training Center. Experienced professional instructors and hands-on computer time for students combine to make the training experience an invaluable asset for your operation.

Full descriptions of the courses offered are given in the HP 3000 Specification Guide.

SOFTWARE SUPPORT SERVICES

A well-defined set of software support services is offered with HP 3000 computer systems. Since customer support needs can differ considerably, these support services are available to provide a flexible range of support. The software services are divided into two main categories:

- **CSS—Customer Support Service,** for customers who choose a close support relationship with Hewlett-Packard.
- **SSS—Software Subscription Service,** for customers who prefer to rely on their own resources for software support.

Documentation Distribution Services

In addition to the software support services described above, Hewlett-Packard offers two types of Documentation Distribution Services that are appropriate for customers with a large programming staff who wish to be individually informed of software problems or keep their documentation up to date. These services are:

- **SNS—Software Notification Service,** provides one copy of the Software Status Bulletin and the Communicator
- **MUS—Manual Update Service,** provides one copy of updates or new editions to manuals automatically whenever they are issued. The various sets of manuals that can receive this support are specified in the Price/Configuration Guide.

The reference sheets in the HP 3000 Specification Guide provide a full description of the Software Support Services outlined above.

HARDWARE SUPPORT SERVICES

A range of maintenance service tailored to your needs is available to ensure that your system runs smoothly and reliably. The services provided by the Customer Maintenance Agreement provide preventive maintenance visits and emergency repairs.

For the first 90 days after installation, an on-site warranty provides parts and labor. After the warranty period, service is continued under the Customer Support Services Agreement. The cost is determined by your configuration and the type and frequency of service best suited to your company's needs.

For more information on Hardware Support Services, refer to the reference sheet in the HP 3000 Specification Guide.

HP GENERAL SYSTEMS USERS GROUP

The Users Group is an independent world-wide organization for the purpose of exchanging techniques and ideas among HP 3000 users. Hewlett-Packard's Customer Relations Manager works closely with the Users Group to promote communication between Hewlett-Packard, the General System Division in particular, and HP 3000 users. Membership is open to any interested individual using an HP 3000. You will find that meeting with other users and exchanging ideas provides a stimulating environment in which to sharpen your programming and operational techniques.

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

Chapter 2

TRANSACTION PROCESSING

CREATING A TRANSACTION PROCESSING SYSTEM
PRODUCTION TOOLS: FUNDAMENTAL OPERATING SOFTWARE
IMAGE/3000
QUERY/3000
KSAM/3000
HP V/3000
UTILITIES
PROGRAM EXECUTION FACILITY
DEVELOPMENT TOOLS: LANGUAGES
DEBUGGING AIDS
COBOL II/3000
RPG/3000
FORTRAN/3000
BASIC/3000
SPL/3000
APL/3000

Today, many businesses are moving away from large batch processing computer systems toward networks of smaller computer systems running innovative on-line transaction processing applications. These installations are typified by a relatively large number of users who communicate with the computer system through terminals and who present the computer with an uneven processing load, heavy terminal and disc I/O demands, and the need to share code and data among the many users. The HP 3000 computer system is designed to meet the needs of such a transaction processing environment.

In a general sense, the expression "transaction processing" can be applied to virtually any interaction between a computer system and its users. A transaction is a series of steps in a predefined logical sequence which accepts input, performs some sort of data processing or manipulation, and generates output. The term "transaction" implies that there is an exchange taking place, and this is certainly true of an interactive transaction processing system where transactions are used to access information stored in a computer's data bases and files.

There are two major advantages of an interactive transaction processing system:

- It allows the entry, manipulation and retrieval of data at various dispersed locations—usually the locations where the data is actually generated and used for business decisions. This allows the people who are most familiar with the data to enter and interact with it.
- It speeds up the business cycle. Rapid and accurate data input and retrieval can be performed by relatively inexperienced personnel to make up-to-date information constantly available. This results in faster response to customer demands, thereby providing a major competitive edge.

CREATING A TRANSACTION PROCESSING SYSTEM

The creation of a transaction processing system involves the skillful blending of user requirements and system capabilities. First, the needs of all system users from data entry personnel, through system management and corporate staff are identified. Next the system designer or analyst translates these needs into logical specifications based on the resources and capabilities of the computer system. Finally, the programmer interprets the specification into a language the computer can understand.

In effect, four sets of requirements must be examined and met in the creation of a transaction processing system: those of the user, the system designer, the programmer, and the computer system itself.

The User:

- wants to see a conversational interface to the application program
- wants the ability to provide one-time reports without additional programming effort
- doesn't want to get involved in learning the computer system software.

The system designer:

- needs to determine the system requirements of the end user
- needs to implement three important aspects of a transaction processing system—fast-response time, rapid information access, and high transaction throughput
- should be provided with design aids to help his analysis of the application needs.

The programmer:

- needs to be familiar enough with the host computer to translate the system designer's specifications into an actual application program
- should be provided with system characterizations which define performance considerations and design tradeoffs of the computer system
- should be provided with programming tools which increase his efficiency in writing, debugging and maintaining the application program.

The computer system:

- must provide hardware and software which operate well in a transaction processing environment
- must provide all of the design aids, system characterizations, software development tools and system software which will allow the system designer and programmer to design and implement a successful transaction processing system.

Transaction Processing on The HP 3000

There are several approaches that the designer of an HP 3000 based on-line transaction processing system may use to facilitate user interaction with the system. In the most commonly used approach, the system is designed so that each user initiates a separate session and invokes an application program with a RUN command, as illustrated in Figure 2-1.

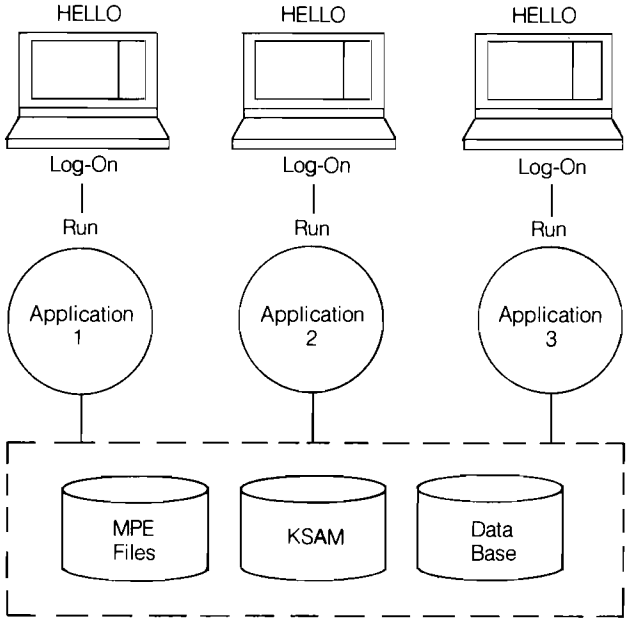


Figure 2-1. On-line transaction processing

With an alternative approach, which also runs one process per terminal, the user is completely isolated from the HP command language. As illustrated in Figure 2-2, a single process (father) manipulates multiple processes (sons), which have no connection with the MPE command language during their execution. Each son process opens a terminal and prompts the user to perform a specific transaction via a menu or screen display. The terminal user simply enters the required data, without ever having to issue a RUN command. When the user is finished with the application, the son process relinquishes control of the terminal, which can then be used in normal interactive mode.

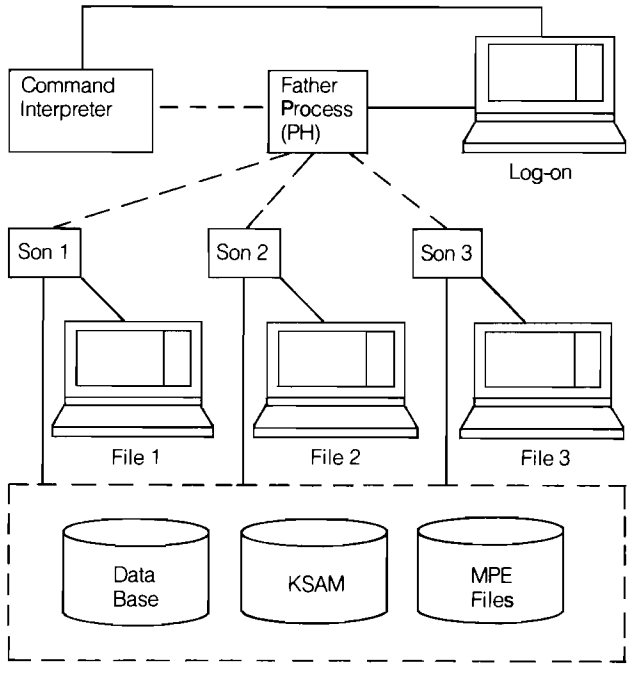


Figure 2-2. Alternative approach

PRODUCTION TOOLS: FUNDAMENTAL OPERATING SOFTWARE

A transaction processing system must achieve good system throughput and resource utilization. The design, development, and maintenance of the programs should be relatively simple and straight-forward. The HP 3000 is equipped with a complete set of tools, depicted in Figure 2-3, which meet the needs of a transaction processing system and make the implementation of that system an easy task.

Every HP 3000 is provided with Fundamental Operating Software (FOS) consisting of:

- MPE Operating System and File System
- IMAGE/3000: a comprehensive data base management system
- QUERY/3000: an English-like inquiry facility used with IMAGE
- KSAM/3000: a keyed sequential file access method
- HP V/3000: a self-contained data entry system and a programmatic terminal interface
- Utilities for system administration and control
- Facility to execute compiled programs without the source language compiler

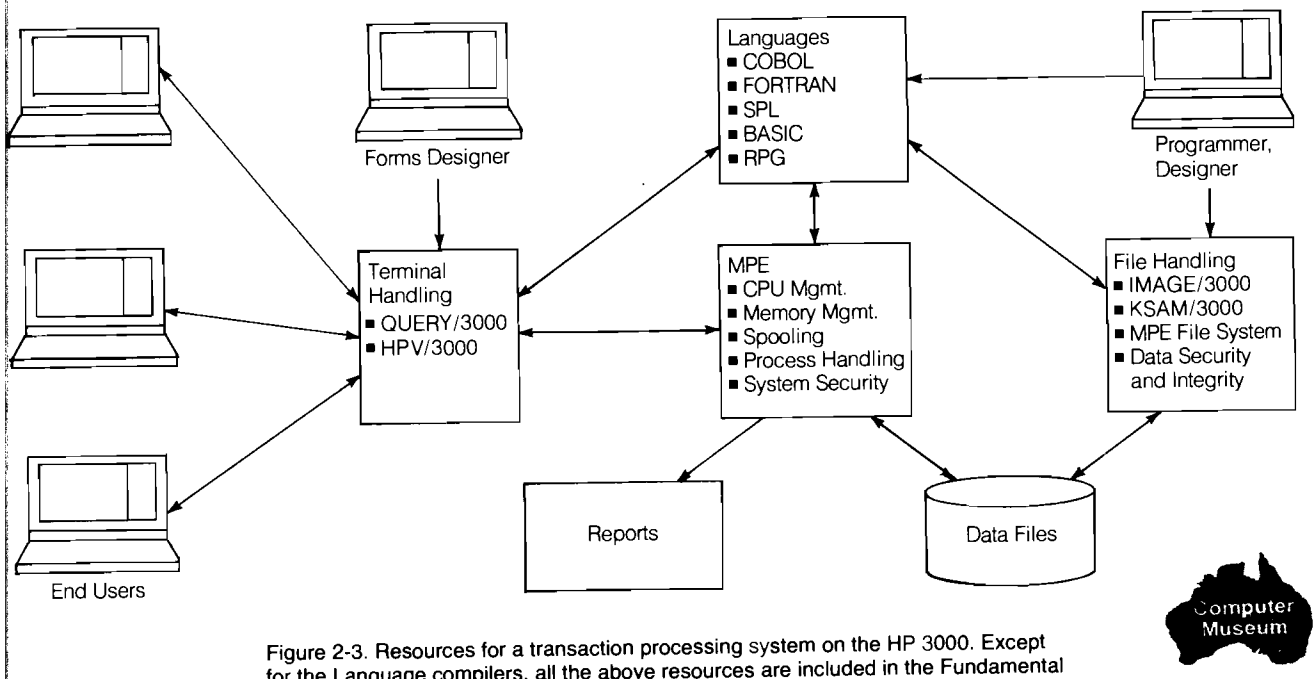


Figure 2-3. Resources for a transaction processing system on the HP 3000. Except for the Language compilers, all the above resources are included in the Fundamental Operating Software that is part of every HP 3000 computer system.

The Fundamental Operating Software equips the HP 3000 with the functions it needs to be a ready-to-run application system. FOS includes a complete set of high level procedures for data entry and data management, callable from application programs. It also provides many interactive facilities to terminal users for screen design and data access.

With the addition of a language compiler, FOS becomes a powerful program development system. Programs developed and compiled on one HP 3000 execute on any other HP 3000 without installing any software other than the program itself.

In addition, HP 3000 transaction processing systems can make use of the Hewlett-Packard Distributed System Network (HP-DSN), a powerful tool for collecting, reporting, and sharing data among physically remote locations, as described in Chapter 4.

IMAGE/3000

IMAGE/3000 is a powerful data base management system. Included in the Fundamental Operating Software, it handles multiple files and helps you define and create a data base tailored to your special requirements. IMAGE is especially designed for data handling situations where a large number of files are needed and the inter-relationships of data within the files are complex.

A data base is a collection of logically-related files containing both data and structural information. Pointers within the data base allow you to access related data and to index data across files.

The primary benefit of the IMAGE data base management system is the time savings which result from the file consolidation and the ease with which programs are developed and maintained.

Most information processing systems that serve more than one application area contain duplicate data. The consolidation of multiple files into a single data base eliminates most of the data redundancy. Through the use of pointers, logically related items of information are chained together, even if they are physically separated, allowing the data to be used by any program needing it. Since there is only one record to retrieve and modify, the work required for data maintenance is greatly reduced, and all reports using those items of information are consistent.

IMAGE allows the data base to be structured independently of the application programs that use it, so that programs only need to define those data items actually used, rather than the entire file. As a result, changes can be made to other parts of the data base without modifying the program that accesses the data base. Coding can begin before the data base structure is finalized, as long as the format of the individual data items accessed by the program is known. Future enhancements to the data base can be implemented without impacting existing programs. The logical separation of the structure of the data base from the applications results in greater flexibility and substantial time savings for both the data base designer and the programming staff.

Data Base Structure

IMAGE is primarily a path oriented or chained approach to data retrieval. Pointers are maintained which logically connect those records with common attributes into chained lists. This allows cross-referenced access to collections of data down to the smallest unit and makes it possible to access related data quickly.

An IMAGE data base consists of data items, data entries, and data sets. A data item (field) is the smallest accessible data element. It consists of a value referenced by a data item name. In general, many data item values are referenced by the same data item name.

A data entry (record) is an ordered set of related data items. The order of the data items within a particular data entry is specified when designing the data base. The length of a data entry is the combined lengths of all data items within it.

A data set (file) is a collection of data entries. Each data set is referenced by a unique data set name.

A data base is a named collection of related data sets. It is defined in terms of data items and data sets. A data base may be defined with up to 255 data item names and 99 data set names.

An IMAGE data set is either a master or a detail data set.

Master data sets:

- are used to keep information relating to a uniquely identifiable entity; for example, information relating to a customer (such as account number, name, address, and credit rating)
- allow for rapid retrieval of a data entry since one of the data items in the entry, called the search item, determines the location of the data entry
- can be related to detail data set containing similar search items and thus serve as indexes to the detail data set.

Detail data sets:

- are used to record information about related events; for example, information relating to each sale for each customer in the master data set
- allow retrieval of all entries pertaining to a uniquely identifiable entity; for example, an account number may be used to retrieve all data entries containing information about sales to that customer.

As illustrated in Figure 2-4, each master data set may serve as an index to a maximum of 16 detail data sets, and each detail data set may be indexed by up to 16 master data sets (or none).

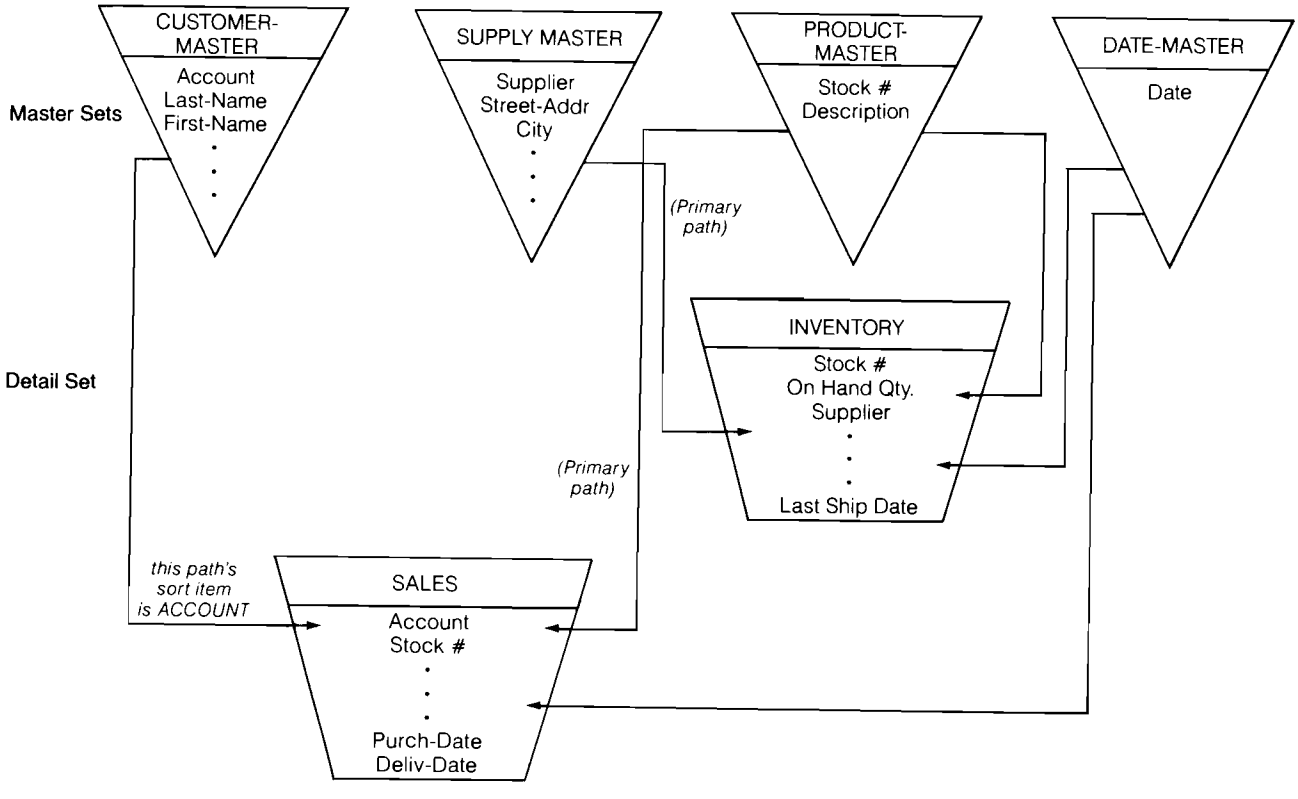


Figure 2-4. Example of master-detail data set relationships.

Accessing the Data Base

You access a data base through call statements to IMAGE Data Base Management Subsystem (DBMS) procedures in COBOL, FORTRAN, BASIC, or SPL programs, or through compiler generated calls in RPG programs. These options give you the ability to select the language most appropriate to each application. The IMAGE procedures locate data, maintain pointer information, manage the allocated file space, and return status information to you. The tasks performed by the IMAGE procedures relieve you of the "bookkeeping" normally associated with file management and allow you to concentrate on applying the processing power of whatever programming language you are using.

Some of the functions performed by IMAGE procedures are:

- Adding a new entry to a data set.
- Deleting an entry from a data set.
- Reading some or all of the data items of an entry.
- Changing the values of data items of an entry.

There are four modes of access available:

Serial—in this mode IMAGE starts at the most recently accessed record and sequentially examines successive records without regard to their key value. Forward and backward serial access is available.

Directed—in this mode the calling program specifies the record address of the data entry from which the desired data items are to be retrieved.

Calculated—this type of access involves retrieval of master data set entries based upon the search item value.

Chained—this type of access consists of successive retrieval of all entries in a detail data set which contain the same search item value.

With Distributed Systems/3000 (DS/3000) software, you may also access a data base on a remote HP 3000. The same call statements used to access a local IMAGE data base can be used with DS/3000 to access remotely located data bases across a communication line. Both the local and remote HP 3000's must have DS/3000 software.

Data Base Security

In addition to the security safeguards of MPE, IMAGE provides the data base administrator with further protection for the data base. The key to the strength of IMAGE/3000 security is the ability to control access not only to specific data sets (files), but also to each data item (field), as shown in Figure 2-5. You can define up to 63 user classes, each with a related password, and associate each class with either read or write access to specific data sets and items.

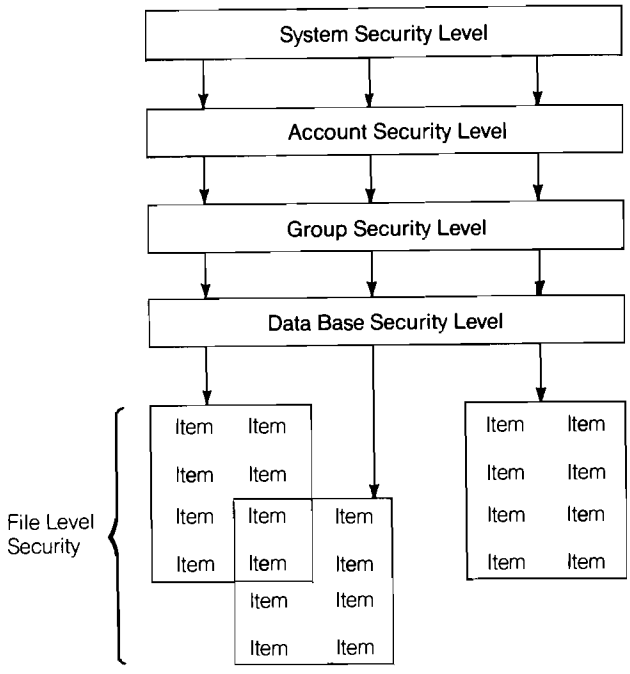


Figure 2-5. Total system and data base security

Therefore, the data base is protected down to the smallest unit of information, for example, a data item specifying an employee's pay rate. This ensures that the only users who access and/or change specific items are those whose job functions require it.

Because multiple users can be simultaneously accessing and updating an IMAGE data base, data integrity must be assured. You may specify the type of access you need—e.g., read, modify, update, etc., directly from the terminal or through a program. IMAGE will allow that access unless it conflicts with the functions already being performed by the other concurrent users. This approach results in compatible sharing of the data base and ensures data integrity.

Transaction Logging and Recovery

IMAGE/3000 offers a logging and recovery system which is based on MPE III user logging and is designed to restore data bases to a consistent state, both logically and structurally.

The IMAGE/3000 logging facility provides the means to log data base transactions to a log file on tape or serial disc. The log file is a record of all modifications to data base items, including information about previous entries as well as the current state of the data base. User text can be logged in order to facilitate future access and interpretation of the log files; it may also be a useful tool for auditing. The recovery system reads the log file to re-execute transactions against a data base back-up copy in the event of a system failure.

The data base administrator is responsible for enabling or disabling the logging and recovery processes, making logging a global function controlled at the data base level rather than at the individual user level.

Query/3000

QUERY is a companion data base inquiry facility for IMAGE and is also part of the Fundamental Operating System. A self-contained language, QUERY provides a simple means of accessing the data base through the use of English language key words and other character strings. You communicate with QUERY through 17 unique commands to store, modify, retrieve, and report on data in an IMAGE data base. Commands can be entered either from an interactive terminal or a batch input device such as a card reader.

With Distributed Systems/3000 (DS/3000) software, the same QUERY commands used to access a local data base may be used for direct access of remote HP 3000 data bases, across a communications link.

Applications

Since QUERY is designed for the non-programmer, it can be employed in a variety of applications after only minimal training. Both novices and experienced programmers find it extremely valuable. Some of the major application areas include:

- Casual Inquiry of the Data Base—facilitates searching a data base for information without writing a program or waiting for a periodic batch run.
- On-line Data Updates—permit modification or deletion of data on-line, directly from a terminal to the IMAGE data sets (files).
- Report Generation—formats reports with header and column labels, page numbers, group labels and summary statistics. Pre-defined report formats can be catalogued in the system to aid inexperienced users.
- Application Program Debugging—aids in program development. QUERY can be used to build test data as well as to interrogate the results of program and system tests. This feature eliminates the requirement that file-related programs be completed before meaningful functional programs can be written.

Features

In addition to the applications listed, QUERY performs several other valuable functions.

- Selection of Data Through Logical Comparisons—locates specific entries for processing based on logical criteria specified in a FIND command.
- Creation and Storage of Procedures—stores frequently used or lengthy commands as individual procedures in a command file, avoiding the necessity of retyping them when needed.
- Display Data Base Structure—displays structural information about the data base and shows the relation between data items and the data sets in which they are located.

Security Provisions

QUERY adheres to all of the security provisions included in the IMAGE data base. After QUERY is invoked, a security password must be entered which determines which data entries and data items may be accessed.

Data Types

QUERY manipulates many of the various IMAGE data types. Each specific data type has a length ranging from two digits plus a sign to 132 characters. The types of data are:

- One-word integer numbers
- Two-word integer numbers
- Two-word real numbers
- Four-word real numbers
- One-word logical values as absolute numbers
- Zoned decimal numbers
- Packed decimal numbers
- ASCII character strings
- ASCII character strings containing no lowercase alphabetics
- One-word integer numbers corresponding to COBOL computational data
- Two-word integer numbers corresponding to COBOL computational data.

Inquiries

The FIND command locates entries in a data base. Logical selection criteria, included in the command, allow only the entries pertaining to a given inquiry to be returned. Up to 50 logical relationships can be specified in one command.

Reporting

Data entries can be displayed or reported according to a user-specified format. After the data records have been located by the FIND command, a REPORT command is used to specify which items within those records QUERY is to display. REPORT also specifies such items as titles, column headings, page numbers, and line spacing. Figure 2-6 illustrates a QUERY report.

Updating

A data base may be updated by adding, deleting, or modifying an entry using the UPDATE command which operates within the limits established for a password. QUERY prompts for the values needed to complete a transaction.

```
>FIND VENDOR IS "AJAX" AND ORDER-STATUS IS "O"

3 ENTRIES QUALIFIED

>LIST ORDER-NUMBER,QTY-LEFT-TO-REC FOR VENDOR IS "AJAX"

ORDER-NO   QTY-LEFT

TESTPO          10
PO-995         875
PO-995        5000
```

You can obtain a simple report by merely using a LIST command.

```
>REPORT
>>H1,"OPEN PURCHASE ORDERS FOR AJAX",60
>>H3,"PART-NUMBER",14
>>H3,"ORDER-NUMBER",32
>>H3,"QUANTITY",45
>>H3,"DUE-DATE",50
>>H3,"CONTROLLER",70
>>S1,DATE-DUE
>>D1,PART-NUMBER,18
>>D1,ORDER-NUMBER,30
>>D1,QTY-LEFT-TO-REC,44
>>D1,DATE-DUE,57
>>D1,CTLR,65
>>END

                    OPEN PURCHASE ORDERS FOR AJAX

   PART-NUMBER    ORDER-NUMBER    QUANTITY    DUE-DATE  CONTROLLER
200000            TESTPO               10      042880          99
AH63              PO-995              875      071580          40
300001            PO-995             5000      072680          99
```

To request a more complex, formatted report, you can issue a REPORT command and specify headings (H1, H3), detail lines (D1) and sort sequence (S1).

Figure 2-6. Typical QUERY report procedure

KSAM/3000

KSAM/3000 (Keyed Sequential Access Method) is a part of Fundamental Operating Software that provides sophisticated file access by key values within the data record. Each data record contains one primary key field and may include up to 15 alternative ones. Access to these records can be sequential or random by either primary or alternative key value. KSAM also supports key access by physical or logical record number, or by chronological order.

You may use KSAM to retrieve data with part of a key value rather than the entire key. Called partial or generic key search, this approach is ideal for values that share a common beginning. Suppose you wish to find all the customers in a certain geographic area. By specifying only the common first three characters of the zip code—like 950, it is possible to read quickly 95050, 95060, 95065, etc. Records in a KSAM file can also be located through approximate searches. For instance, you can retrieve the first record whose key value is greater than or equal to "01/01/79".

If you have existing indexed sequential files, KSAM/3000 can facilitate their conversion to the HP 3000. For example, RPG programs on an IBM SYSTEM/3 using indexed access method require no coding changes to run on the HP 3000.

File Structure

A KSAM file consists of two physical disc files—one for data and one for keys. The data file contains all the data records while the associated key file holds one or more sets of entries that maintain the primary and alternate logical sequences of the data records. When constructing a file, key entries are dynamically added in ascending order. Any key file restructuring required to accommodate additional data file records is performed automatically by KSAM in an operation that is invisible to you. Figure 2-7 illustrates a KSAM file with a primary key and one alternative key.

Data fields, including key fields, may contain these types of data:

- signed binary integer,
- double-word signed binary integer,
- real,
- four-word long real,
- ASCII character string,
- packed decimal,
- zoned decimal.

Features

KSAM/3000 provides a number of features, including:

- Multiple Keys—One primary key and up to 15 alternate keys may be specified for any KSAM data file. Each key is ordered in sequence by its value with no relation to other keys.
- Duplicate Key Values—While KSAM normally expects unique values for keys, some key types (such as zip codes) may logically contain duplicate values. You may optionally define these keys to allow non-unique data.

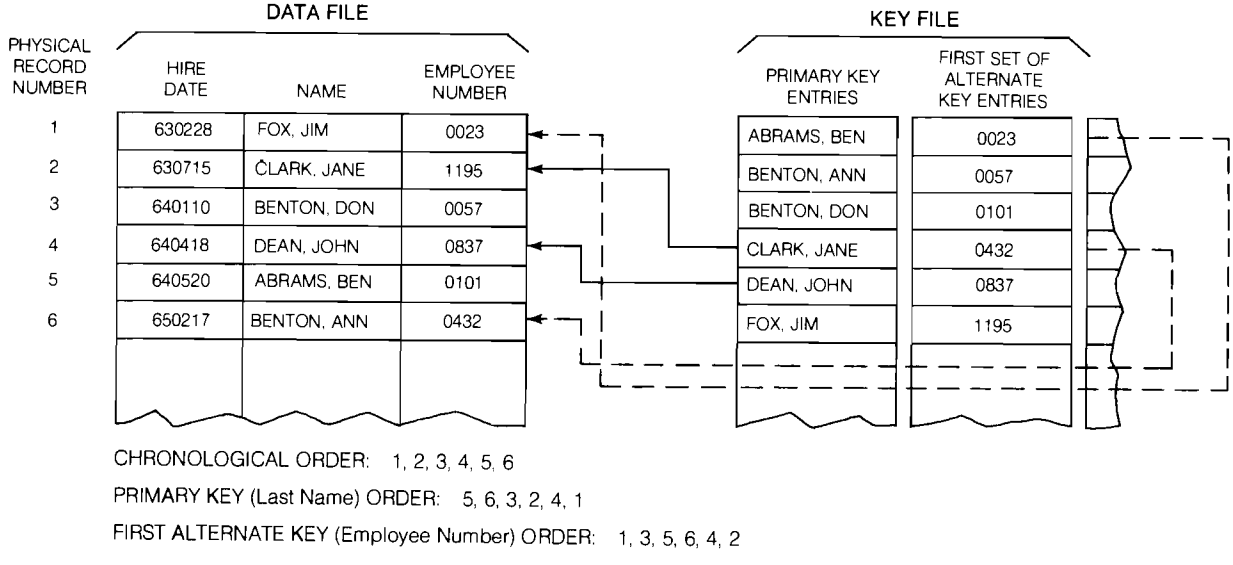


Figure 2-7. KSAM file example.

- Retrieval by Generic Key—You may elect to retrieve records based on a match with the first part of the key only. For example, all stock numbers for similar parts can be read if they share a common prefix.
- Retrieval by Approximate Match—Records may be retrieved based on their relation to a specified key value such as all dates greater than 770401.
- Fixed or Variable Length Data Records—For added flexibility, both fixed and variable length record formats are allowed for KSAM files.

These features make KSAM an appropriate tool for applications where the variety of file access options is more important than flexibility in data base design.

HP V/3000

V/3000 is a comprehensive software subsystem of the Fundamental Operating Software that implements and controls source data entry, and also provides an interface between terminals or files, and any transaction processing application.

Source Data Entry

As a source data entry system, V/3000 provides easy forms design with data editing and validation built into the forms. It also provides a ready-to-use data entry program so that you can enter data into the system with no programming effort. This same program allows you to "browse" the entered data and modify it as it is entered.

Thus source data entry through V/3000 can be done without the use of a compiler. If, however, you need additional or different capabilities, you can use the V/3000 procedures to modify the existing data entry program or write your own application.

Programmatic Interface

As an interface to transaction processing applications, V/3000 provides a set of procedures that allow you to control terminals, forms, and files from an application program. These procedures are available to programs in COBOL, RPG, FORTRAN, BASIC, and SPL.

V/3000 also provides a reformatting capability. You can enter specifications to control how data in a batch file is to be reformatted, and then run a program to actually reformat the data.

The V/3000 procedures and the reformatting capability, either singly or in combination, provide a "front end" to existing transaction processing applications. Thus, V/3000 allows you to concentrate on processing problems rather than on data editing or interface to terminals.

A combination of menu type screens and terminal function keys assist you in manipulating forms design. For example, with the FORMSPEC facility you can design forms, change existing forms, add new forms, and delete forms or fields as shown in Figure 2-8.

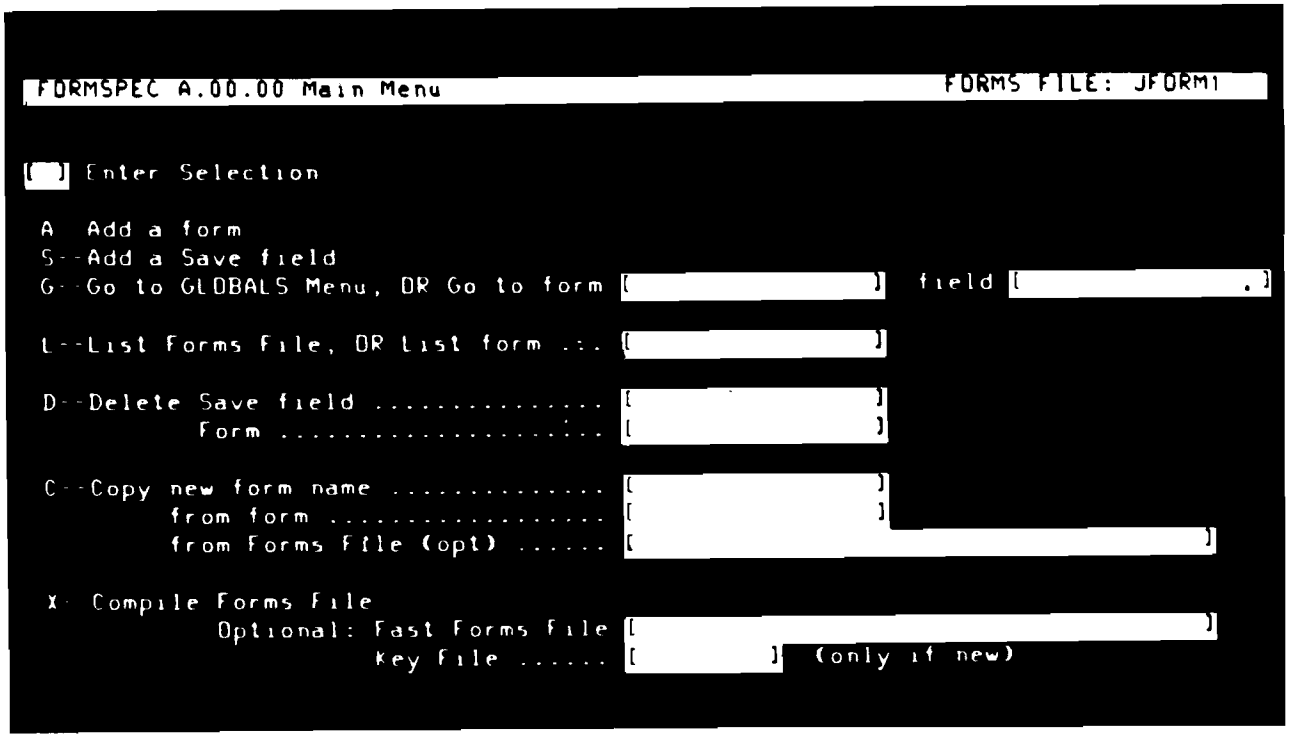


Figure 2-8. Main menu screen FORMSPEC facility.

Features

The following are the main features of the V/3000 system:

- A forms design program (FORMSPEC)—allows quick and easy forms design at a terminal using "menus."
- Advanced forms design (through FORMSPEC)—provides data editing, formatting, and validation of data as it is entered.
- A data entry program (ENTRY)—provides immediate data entry and modification with no programming effort.
- A flexible data reformatting design program (REFSPEC) specifies reformatting of data to suit an existing application.
- A batch program (REFORMAT)—reformats the data according to the REFSPEC formatting specifications and writes it to a file for use by an application.
- A set of procedures that provide a powerful language interface to terminals, forms, and files from user programs written in COBOL, RPG, BASIC, FORTRAN, or SPL.

Designing Forms

The FORMSPEC program allows you to design forms in as simple or as complex a manner as you desire. Form design with FORMSPEC can be thought of as having four levels of complexity:

- ASCII Collection—You draw the form on the terminal screen and accept any ASCII data entered by the operator.
- Simple Editing—You draw the form on the screen and specify edits based on field type (optional, required, display only) or data type (character, numeric, or date). No special language is required for these edits.
- Full Field Edits—You specify a full range of field edit statements that apply to individual fields in a form. These include: minimum length, range checks, and pattern checks. A subset of the FORMSPEC field specification language is used for these edits.
- Advanced Processing—You specify movement of data between fields and forms, formatting of data (JUSTIFY, FILL, STRIP), alteration of forms sequence, and conditional processing based on the result of editing statements. This level uses the full range of the FORMSPEC language.

Once designed, forms are stored in a forms file and compiled for use in data entry. Forms can be modified during initial design, or even after the forms file is compiled.

Data Entry Program

A stand-alone data entry program, ENTRY, displays forms at the terminal, accepts data entered on the forms, and writes the data to a batch file.

ENTRY operates in two modes: Data Collection and Browse/Modify. The mode when ENTRY is first run is always data collection; the operator must request browse/modify by pressing a terminal function key. As indicated by their names, data collection mode is used to collect data from the terminal, and browse/modify to look at the collected data and modify it if necessary.

In Data Collection mode, a set of terminal function keys allows the operator to:

- Request the first (or head) form.
- Terminate a repeating form and display the next different form.
- Clear the current form to its initial values.
- Print the current form with its initial values on the line printer.
- Request browse/modify mode to view and/or change data already written to the batch file.
- Terminate the ENTRY program.

In Browse/Modify mode, a set of terminal function keys allows the operator to:

- Display data from the first batch record on its form.
- Display data from the previous batch record on its form.
- Display data from the next batch record on its form.
- Clear the current form to the values displayed before any current modifications were entered.
- Delete the data currently displayed at the terminal (in effect, this deletes the current batch record).
- Print the current record, before modification, on the line printer.
- Continue entering data after interrupting data collection.
- Terminate operation of ENTRY.

Reformatting Data

Data entered through V/3000 forms is usually written to a batch file. This file can then be used as input to an application program. Sometimes it is necessary to reformat the data in the batch file so that it meets the input requirements of the application program. V/3000 meets this need with a reformatting capability that allows you to:

- Combine data entered on several forms into one output record.
- Separate data entered on a single form into several output records.
- Rearrange data within a record, inserting constants, and generating check digits before writing it to the output file.
- Format data within fields by justifying, filling, stripping characters, or adjusting the sign of a numeric value.

Program Interface

V/3000 provides a library of high-level procedures for use by all terminal-oriented applications. These procedures can be called by RPG, COBOL, BASIC, FORTRAN, or SPL programs. The program interface provides:

- Terminal Interface—Procedures to open and close a terminal file, to display a form at the terminal, and read data entered in fields of the displayed form.
- Forms File Interface—Procedures to open and close a forms file, to get the next form in sequence from the forms file, and to print the current form, with its contents, on the line printer.
- Data Manipulation—Procedures to initialize a form to its initial values, to perform any FORMSPEC or user-defined edits, and to perform any final form processing as defined by FORMSPEC.
- Data Entry—Procedures that open and close a batch file, write data to a batch file, or read data from the batch file.
- Access to Data—Procedures to read entered data to a program buffer, or write data from a program buffer; data from an entire form can be read or only from selected fields; similarly, data can be written for an entire form or only for selected fields. Data passed to or from fields can be converted to or from a variety of data types.
- Status/Error Control and Custom Error Messages—Procedures to set error flags, display custom error messages, and enhance fields in error.

Utilities

Program development on the HP 3000 is made easy through a powerful editor and other commonly-used utility routines.

EDIT/3000

EDIT/3000, the HP 3000 text editor, allows you to create and manipulate files and upper- and lower-case ASCII characters with great ease. Lines, strings, and individual characters can be located, inserted, deleted, and replaced. The files to be edited may contain source language programs or text material such as reports.

You interact with EDIT/3000 through a set of editor commands that includes commands common to many other text editors throughout the computer industry. As illustrated in Figures 2-9 and 2-10, experienced users can write programs directly on the terminal, bypassing both coding sheets and punched cards.

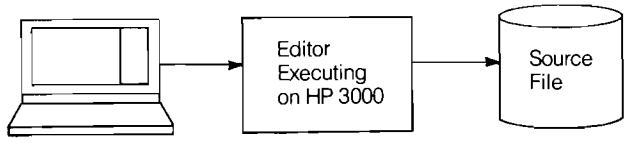


Figure 2-9. Creating a source file at your terminal

After initiating a session with the :HELLO command, you can use the Editor to create a source file. Figure 2-10 is an example of the simplest method:

```
:EDITOR ———————————————————— Initiate Editor Execution

HP32201A.6.01 EDIT/3000 WED,DEC 7,1977, 2:00 PM
(C) HEWLETT-PACKARD CO. 1976
/SET FORMAT=COBOL —— Put Editor in COBOL mode. Enter ADD command and enter lines of source code.
/ADD                 Space to column 2 (which is COBOL column 8) to enter statements.
    1      $CONTROL USLINIT,MAP
    1.1     IDENTIFICATION DIVISION. ———— Space to COBOL column 8
                                          (which is Editor column 2)
                    .                     to enter COBOL statements.
                    .
                    .

    6.7             STOP RUN
    6.8    // ———————————————— Enter // or Control Y to stop adding lines.

/LIST ALL ———————————————————— List code and check it.
    1      $CONTROL USLIMIT,MAP
    1.1     IDENTIFICATION DIVISION.
                    .
                    .
                    .
    6.7             STOP RUN

/KEEP SOURCEX ———————————————— Store source code in a file named SOURCEX.
/LIST ALL, OFFLINE ——————————— Obtain a printed listing of the source code.
***LISTING COMPLETED***
/EXIT
```

Figure 2-10. Using the HP 3000 text editor

Basic editing functions such as adding, locating, changing, deleting and listing lines are provided using single-word commands (which can even be abbreviated to a single, unique letter), whether the edits are applied to one line or many lines. Most functions operate either on specified line numbers or on lines containing specified character strings. Groups of lines may be moved or copied within a file or from one file to another.

EDIT/3000 lets you write complex edit command sequences where editing is based upon conditions found within the text itself. In such command sequences the editor language assumes an ALGOL-like structure with the commands WHILE, NOT, and OR acting upon statements that can be simple or compound in nature.

Through the use of the Z::= and USE commands, you can store sequences of editor commands in the system and then invoke them at will to act upon text in the editor work file. In this manner, commonly-used sequences of editor commands, such as complex WHILE sequences, can be activated with a single USE command rather than by reentering the entire sequence.

A very powerful feature of EDIT/3000 is that it allows you to write procedures in COBOL, FORTRAN, or SPL that perform editing functions not otherwise possible with the standard repertoire of editor commands. You write, prepare, and store your procedures on disc and then invoke them with the PROCEDURE editor command. The editor passes the lines of text specified in the PROCEDURE command, one at a time, from its work file to the designated procedure, which then manipulates them and passes them back to the editor. EDIT/3000 substitutes the modified lines for the original ones in the work file. After the procedure is finished, the editor resumes processing standard editor commands. This feature allows you to perform any editing function that can be done a line at a time.

SORT-MERGE/3000

SORT-MERGE/3000, the HP 3000 sort/merge program, allows you to sort records in a file into a prescribed order and to merge records by combining two or more previously sorted files into one sorted file. SORT-MERGE can be used as a free-standing subsystem through a few simple commands in either an interactive session or a batch job. It can also be accessed by programs written in COBOL, FORTRAN, or SPL.

FCOPY/3000

FCOPY, the HP-3000 file copier, may be used for all file copying operations on the HP 3000, including KSAM files. With FCOPY you can transfer files from disc, magnetic tape, or any other valid input device to disc, magnetic tape, or any other valid output device.

Facility to Execute Compiled Programs Without the Source Language Compiler

Included as part of the Fundamental Operating Software of each HP 3000 system is the facility to execute compiled programs without the source language compiler (except for those written in APL\3000). With this facility, programs compiled on one HP 3000 may be run on other HP 3000 computer systems without requiring any additional software other than the program itself. Thus one compiler on one system can supply the program development needs of an entire network.

Many other utilities are included in the Fundamental Operating Software to help administer the HP 3000 resources and assist in program development.

For a more detailed explanation of all HP 3000 utilities, turn to Section II of this manual, where each product is described in a product reference sheet.

DEVELOPMENT TOOLS: LANGUAGES

In addition to the components of the Fundamental Operating Software, six high-level programming languages are available for applications program development. The HP 3000 provides highly compatible interfaces between these languages and the subsystems of the Fundamental Operating Software. Thus, the data entry, data management, and utility subsystems can be combined with one or more of the language compilers to provide a highly sophisticated program development tool.

The six programming languages available with the HP 3000 are:

- COBOL II/3000—ANS COBOL X3.23-1974 with extended features
- RPG/3000—Compatible with the industry standard RPG
- FORTRAN/3000—ANS FORTRAN X3.9-1868 with extended features
- BASIC/3000—Compatible with industry standard with extended capabilities
- SPL/3000—A combination high-level and machine-dependent language especially developed for programming the HP 3000
- APL\3000—Industry standard with significant extensions (available only on the HP 3000 Series III)

Once a program written in any language (except APL\3000) is developed and compiled, it can run on any HP 3000 system equipped only with the Fundamental Operating Software.

Program development on HP 3000 computer systems is interactive and can be done concurrently with production data processing.

HP 3000 languages may be used concurrently in any combination by many users or, for multiple compilations, several languages may be incorporated within a single job or session.

All of the compilers on an HP 3000 are invoked in the same general way. This reduces the amount of training required, since learning to use one compiler is essentially learning to use them all. The HP 3000 also preserves your programming investment by incorporating subroutine call conventions in the machine's microcode. This enables programs in all languages, except APL, to call subroutines written in other languages.

A program written in any of the available programming languages (except APL) may also manipulate an IMAGE data base, access a KSAM file, or use HP V/3000 forms—making the HP 3000 data base management and data entry capabilities accessible wherever they are needed.

All programs which run on an HP 3000 access data through the MPE file system. Thus, the programmer need not be concerned with the physical type of device on which the data resides. No changes need to be made to a program to switch input or output from one device to another, for example, from cards to an interactive terminal. Programs used in batch jobs can also be used without modification in interactive sessions, and vice versa. In addition, files created by one language can be accessed by any other language. See Chapter 5 for more details on MPE and the file system.

DEBUGGING AIDS

The HP 3000 provides a wide variety of aids for debugging programs at many different levels. MPE commands such as SHOWJOB, LISTF, or SHOWOUT are available to inquiry about file status of a job or a file. Compiler subsystem $CONTROL commands may be used to specify whether to print source code, object code, warning messages, or a symbol map. The #IF compiler subsystem command permits conditional compilation of portions of the program, allowing debugging statements to be inserted permanently in the code and compiled only when needed.

Two powerful capabilities available during execution of a program are Debug and Stack Dump. The Debug facility is an intrinsic procedure which provides an interactive debugging facility to enable you to set breakpoints, display the contents of memory locations, modify memory locations containing data, display and/or modify the contents of registers, and so forth.

The Stack Dump facility is composed of an intrinsic that enables a program to selectively dump any part of the stack and a stack analysis mechanism that is activated when a program aborts. If the program is running interactively, an automatic call to the debug procedure is generated. If the program is running in a batch environment, part or all of the stack is dumped.

COBOL II/3000

COBOL is the primary commercial language used on HP 3000 computer systems. The American National Standard COBOL X3.23-1974 has twelve processing modules. Of these, COBOL II/3000 provides a full Level 2 implementation of:

- Nucleus
- Table Handling
- Sequential I/O
- Relative I/O
- Indexed I/O
- SORT/MERGE
- Segmentation
- Library
- Interprogram Communication

In addition, COBOL II has extended capabilities beyond the ANSI74 requirements. As a subsystem on the HP 3000, COBOL II interfaces with the data communication and data base management subsystems, as well as the MPE file system itself. These extensions reduce programming effort and application development time by providing a complete and comprehensive set of on-line application development tools for the COBOL programmer.

RPG/3000

RPG, the Report Program Generator, is a specialized "high level" programming language developed to provide the business community with a convenient means of producing a wide variety of printed reports. RPG can handle jobs ranging from simple tasks such as printing address labels to very complex ones such as an entire payroll process (printing paychecks, payroll registers, and various allied reports). RPG is ideal for producing such documents as inventory lists, billings, invoices, insurance benefit notices, or summaries of sales, profits and losses, and customer transactions. It is also excellent for updating the master files used in producing these documents.

Like all "high level" languages RPG is problem oriented and relatively free of hardware constraints. It is easy to learn, easy to use, and easy to code. It allows you to specify many important operations with a minimum of effort; you merely make simple entries on specially-formatted coding sheets. Because RPG is a standard language available on many different machines the user can submit programs coded in another manufacturer's RPG directly to the RPG/3000 Compiler with little or no re-coding.

FORTRAN/3000

FORTRAN, the FORmula TRANslator, is one of the oldest and most widely used "high level" programming languages. The initial specifications for the language date back to 1954. Although it was originally designed for use in engineering applications FORTRAN is now also used heavily in business environments.

The standard for this language is American National Standard FORTRAN X3.9-1966 as approved by the American National Standards Institute. FORTRAN/3000 is a full implementation of that standard. To provide a more powerful programming language FORTRAN/3000 includes some extended features not covered by the ANSI standard. Some of these extensions are as follows:

- Source programs may be written in free-field as well as fixed-field format.
- Symbolic names may consist of up to 15 characters (instead of 6).
- FORTRAN/3000 programs may call subroutines written in other programming languages (most notably subroutines written in SPL/3000).
- Character-type data may be used to facilitate string manipulation.
- Arrays may have up to 255 dimensions (instead of 3).

BASIC/3000

BASIC, the Beginner's All-purpose Symbolic Instruction Code, is a "high level" programming language developed at Dartmouth College in 1963-64. Originally BASIC was used primarily for numerical work, but the introduction of string and file handling (not originally available) has made it an appropriate language for use in industry, commerce, universities, and research establishments.

In addition to all features commonly found in BASIC interpreters throughout the industry, the BASIC/3000 interpreter contains a number of extended capabilities, some of which are:

- Four numeric data types (real, integer, complex, extended precision).
- Mixed-mode arithmetic.
- Formatted output.
- Program chaining with common storage.
- External subroutine calls.
- Strings and string arrays.
- File creation and purging under program control.

The BASIC/3000 Compiler converts BASIC programs, which have previously been debugged and stored using the BASIC/3000 Interpreter, into a machine-executable form. A compiled BASIC program exists in the system as an actual code segment rather than as data in a data file and can be run under the operating system rather than through line-by-line interpreting by the BASIC/3000 Interpreter.

Typically a compiled BASIC program runs many times faster than when run interpretively. Of course, it should be understood that actual speed improvement depends on the type of program and the resource demands on the operating system. CPU bound programs, for example, will realize a speed improvement on the order of 10 to 30 times faster, with certain extraordinary programs running up to 100 times faster. I/O bound programs will run 1 to 4 times faster.

APL/3000

APL, A Programming Language, is a unique "high level" programming language developed for scientific and mathematical applications. In recent years it has been used in commercial areas such as statistics, modeling, and finance. With APL, the user has the ability to express complex calculations in a simple and concise manner.

In order to create a more complete APL product, Hewlett-Packard has added many significant features to the APL language as follows:

- Virtual Workspaces and arrays—a virtual memory scheme is used which allows extremely large virtual workspaces and array size.
- Dynamic Incremental Compiler—APL/3000 compiles and saves the code necessary to execute an APL function which means that there can be a significant speed-up of execution on subsequent runs of a function.
- Report Generation—The commercial formatter provides a versatile and easy to use function for the design of a wide variety of formats and reports.
- APLGOL—A structured programming extension to APL has been added which uses ALGOL-like control structures, with the power of APL operators. APLGOL makes the program simple to read and, therefore, to support and modify.
- File Handling—APL/3000 has full access to the MPE file system through the use of shared variables, which means that from within APL, you can use disc files, tape files, card readers and other devices. The component file system allows simplified file manipulation.
- Error Handling—Secure application environments can be created with programmatic handling of errors.
- Extended Control Functions—A set of functions is provided in APL/3000, which allows you to organize the relationship between user-defined functions in a flexible manner, obtain access to local variables and branch to labels in other functions.
- Friendly Editor—The APL function editor provides the power of a full text editor so that text can be entered, edited, and converted to a matrix or vector of characters for later use.
- Other powerful features—Procedure Calls, Programmatic Access to System Commands, Distributed Systems Extensions, and Double Word Integers.

Note: APL 3000 is available only on HP 3000 Series III computer systems.

SPL/3000

SPL, the Systems Programming Language, is a language designed by Hewlett-Packard especially for writing systems software for the HP 3000. SPL combines the best features of both "high level" and machine-dependent programming languages—it allows the programmer to write software quickly, easily, and efficiently, while producing object programs with good code compression and efficient execution times.

Because of the inherent efficiency of machine-dependent languages, most operating systems, monitors, compilers, and subsystems are written in these languages. However, because SPL combines the efficiency of machine-dependent languages with the simple structure of "high level" languages, all HP 3000 software packages—the operating system, the language compilers, the system utilities, the data base management programs, and the telecommunications programs—are written in SPL.

A further benefit is that programmers who do not know the details of the HP 3000 architecture can still use SPL because of its high-level nature. Also, commonly-executed routines can be written in SPL and called from COBOL, BASIC, RPG, FORTRAN, or APL programs.

See the product reference sheets in Section II for more complete descriptions of the HP 3000 languages.

Chapter 3

MANUFACTURING ENVIRONMENTS

MASTER PRODUCTION SCHEDULING
ROUGH CUT RESOURCE PLANNING
PARTS AND BILLS OF MATERIAL
ROUTINGS AND WORKCENTERS
MATERIAL ISSUES AND RECEIPTS
INVENTORY BALANCE MANAGEMENT
WORK ORDER CONTROL
PURCHASE ORDER TRACKING
MATERIAL REQUIREMENTS PLANNING
STANDARD PRODUCT COSTING

Materials Management/3000 is a standard application product which helps manage the materials planning and control functions of a discrete manufacturing company. The objective of the product is to minimize inventory investment while maximizing your customers' satisfaction.

As shown in Figure 3-1, Materials Management/3000 is a fully integrated system that addresses the following areas:

Master Production Scheduling—Provides the capability for on-line production schedule development and on-line "What If" simulation.

Rough Cut Resource Planning—Provides the capability to test the master schedule against known resource "bottlenecks" to determine possible resource constraints.

Parts and Bills of Material—Maintains descriptive, cost, and planning information about each part and how they relate to one another in product structures.

Routings and Workcenters—Maintains descriptive, cost, and planning information about each workcenter in a manufacturer's production facility and the routing sequence necessary to build each assembled or fabricated part.

Material Issues and Receipts—Provides the facility for issues and receipts of materials and also maintains the audit trail.

Inventory Balance Management—Maintains information about inventory balances and provides control of cycle counting and management of inventory locations.

Work Order Control—Tracks scheduled receipts and monitors planned issues (allocations) required for assembly. Also maintains and tracks backorders.

Purchase Order Tracking—Monitors scheduled receipts of purchased materials and maintains vendor information. Provides purchase order commitment information.

Material Requirements Planning—Generates the materials plan with recommendations about what and how much material to order and when to order it.

Standard Product Costing—Provides the capability to roll-up current costs and to accurately set standard costs.

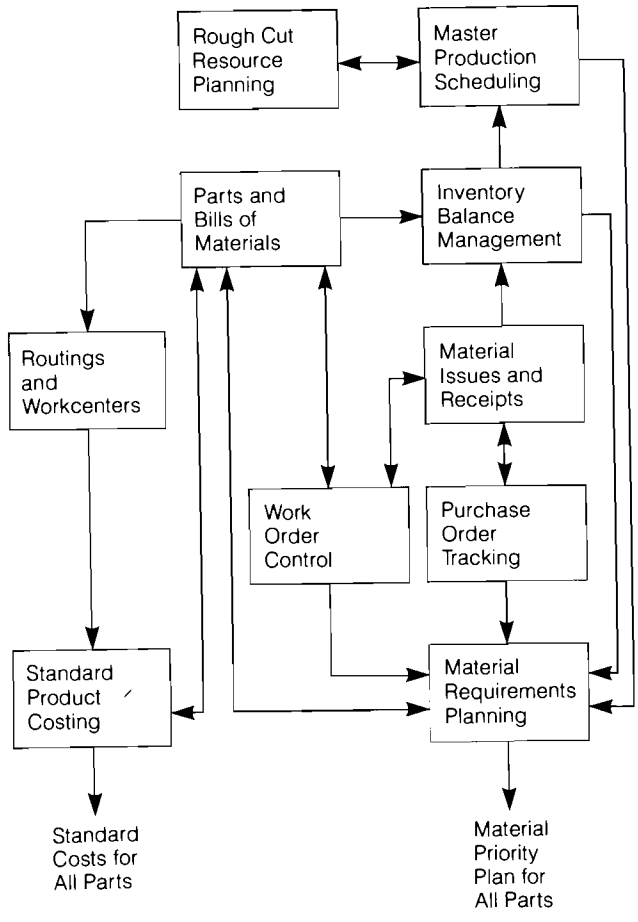


Figure 3-1. Materials Management/3000

Materials Management/3000 is designed to be used with Hewlett-Packard CRT terminals. These terminals have been designed specifically for use in an environment involving on-line processing of information transactions. Special capabilities allow certain functions, normally accomplished by the computer, to be distributed to the terminal. This frees the computer to process a higher volume of user transactions. Terminal memory for lengthy on-line reports, display enhancements for more readable presentations, and special function keys that are programmed by the application are all features that allow your users to increase their productivity. By placing terminals in the users' work areas, information is available when and where it is needed. Data is input by the people responsible for its generation. Timely and complete information for operations and decision making is available within seconds.

The user is guided through each application by a simple menu selection approach. This technique eliminates the need to learn a new language or set of commands. By selecting the function (i.e. retrieve a bill of material, review stock status, review the master schedule, etc.) from the menu and pressing the appropriate predefined terminal function key, the desired information is displayed.

Materials Management/3000 provides a comprehensive set of user customizing features. The application can be used as is or it can be customized, *without programming,* to match the requirements of a particular manufacturing environment. Customizing capabilities include:

- Items can be added to or deleted from the data base and the characteristics of existing items can be changed.
- Menu and transaction screens can be added, deleted, or resequenced.
- Fields can be added, changed, rearranged, or deleted on existing screens.
- System values can be defined and modified.
- Help screens and error messages can be modified.
- Passwords associated with terminals and data items can be defined.
- Arithmetic relationships between data items can be defined.

Materials Management/3000 utilizes Hewlett-Packard's award-winning data base management system—IMAGE/3000, one of the most widely used data base management systems in the world. The advantages of this approach include:

- Elimination of redundant information.
- Capability for on-line update and retrieval.
- Data integrity.
- Predefined manufacturing data base.

In addition, Hewlett-Packard's English-like QUERY/3000 is available to generate reports and retrievals not specifically provided. QUERY/3000 allows a designated user to quickly produce reports without involving a programmer. This increases the productivity of your manufacturing operation by providing quick response for required information.

A significant part of most companies' in-house computer operating expenses are involved with the highly trained personnel necessary to operate a computer system. Materials Management/3000 is designed within an automated operator interface that reduces the training and expertise required to operate the system on a day-to-day basis. The user/operator is virtually insulated from the typical tasks and interactions necessary with most other systems. Operator functions that have been automated include:

- System initialization, start-up, and message monitoring.
- Background job scheduling.
- Tape and disc volume management.
- Handling and recovery of system and job abnormalities.

Hewlett-Packard is committed to providing you more than just software products. We want you to be successful in your system installation. To help you accomplish this goal we offer you, in addition to proven software, a comprehensive training program, detailed user reference manuals, project consulting, and a full support program.

MASTER PRODUCTION SCHEDULING

The Master Production Scheduling module is a management planning and production scheduling tool. It can aid a manufacturer in managing shipment schedules, order backlog, demand forecasts, finished goods inventory, and the mix for product options. Once a master schedule has been generated, a "What If" simulation capability is provided to view the effects of demand and supply changes on the proposed schedule.

Master Production Scheduling Features

- On-line input of production plan, forecasts, and backlog.
- Specification of option mixes.
- Preliminary and final master schedule generation in quantity and dollars, including manufacturing order number assignment.
- On-line "What If" simulation of master schedule.
- Five year planning horizon.

Using Master Production Scheduling

In developing a master schedule, Master Production Scheduling nets existing finished goods inventory and existing master scheduled orders (supply) against forecasts and customer orders (demand) that have been entered through on-line data entry screens. The resulting net requirements are phased into weekly time periods based upon the user's production parameters.

If the demand is greater than the supply, the system suggests rescheduling existing manufacturing orders or starting new orders. For new orders, the system recommends an order number, quantity, and start date. The result of this process is a preliminary master schedule. It can be generated on-line, for a single part, and serves as the starting point for "What If" simulation. The "What If" simulation capability can then be used to simulate the changing of specific items; such as demand, manufacturing orders, start dates, and production rates; in order to view the effect of these changes on the entire schedule.

Once a preliminary master schedule has been approved, it is designated as the new final master schedule and used to replace the old final master schedule. The final master schedule is used as an input to the Material Requirements Planning module.

The final master schedule can be maintained between regular cycles to accurately reflect the current status of production. Information, such as manufacturing order start dates and quantities, can be changed when production realities conflict with the schedule.

Master Production Scheduling can accommodate an unlimited number of options and a varying option mix by manufacturing order. The module will explode end products where options are indicated and develop a master schedule for each option.

ROUGH CUT RESOURCE PLANNING

Rough Cut Resource Planning analyzes the capability of the manufacturing facility to execute the master schedule. It allows management to examine the impact of a proposed schedule on the critical resources which have been defined for each part. The result is a statement of how realistic the master schedule is.

Rough Cut Resource Planning Features

- Identification of critical resources:
  —Labor.
  —Materials.
  —Workcenters.
- Identification of resource requirements by part number.
- Resource loading profiles by part and by resource.

Using Rough Cut Resource Planning

The production and materials departments can develop a profile of the critical resources used at any point in the production process of each master scheduled item. Using this profile and a preliminary master schedule, Rough Cut Resource Planning generates a resource loading report that summarizes the resource requirements of the master schedule. This report highlights the areas where resource constraints will most likely occur. Information is also provided to determine the parts and manufacturing orders which make up the over or under loaded conditions.

PARTS AND BILLS OF MATERIAL

Parts and Bills of Material contains basic documentation about every part in a manufacturer's materials system. This information includes descriptive and planning data, bills of material, standard cost data, and information about engineering changes. This information is used by the other Materials Management/3000 modules.

Parts and Bills of Materials Features

- On-line data base update.
- On-line retrieval of bills of material, where used, and item data information.
- Multiple engineering changes by date or order number.
- Start and stop dates for engineering changes.
- Phantom bills of material for subassemblies consumed in production.
- Option mix planning.
- Standard hard-copy reports.
- Automatic low-level code update.
- Degeneracy checking.

Using Parts and Bills of Material

In a typical installation, interactive terminals are placed in all areas that require bills of material and engineering documentation. Requests for documentation are satisfied directly by on-line retrievals at the terminals, which are located, perhaps, in production engineering, R&D, production control, manufacturing specifications, and other appropriate departments. Requests for lengthy reports entered by the user are serviced in regularly scheduled background runs.

Data entry, editing, and data base update using on-line terminals can be accomplished either by a central group responsible for manufacturing documentation or by each individual responsible for specific portions of the data. Production control, for example, might be responsible for the content of lead time fields, while production engineering might control bills of material.

ROUTINGS AND WORKCENTERS

This module maintains all information necessary to define standard routings (bills of labor). Information about each fabricated part's production requirements is included in its standard routing. Alternate routings may also be defined. The manufacturing facility is defined in terms of workcenters and workstations, identifiable work areas that have specific functions, capacities, and costs.

Routings and Workcenters Features

- On-line data base update.
- On-line review of routing and work center information.
- Standard and alternate routings.
- Capacity specifications of workcenters.
- Cost specifications of workcenters.
- On-line and hard-copy routing reports.

Using Routings and Workcenters

Once the routings and workcenters have been defined, production control can use the information maintained by this module to help in standardizing the documentation that is used during the actual manufacturing of fabricated parts. The information maintained by this module is also an important input to the standard costing process.

MATERIAL ISSUES AND RECEIPTS

The Material Issues and Receipts module offers the capability to control stockroom inventory movement. All transactions involving issues and receipts to the stockroom are recorded and processed here. All record keeping and updating is done on-line, by inventory location. An audit trail of all inventory transactions is automatically maintained.

Material Issues and Receipts Features

- On-line inventory transaction processing.
- Automatic creation of backorders.
- Filling of backorders in priority sequence at time of receipt.
- Generation of picking lists.
- Generation of material receipt document.
- Issue by exception.
- Maintenance of the audit trail.

Using Material Issues and Receipts

This module is used to receive material against work orders and purchase orders entered by other Materials Management/3000 modules. A terminal in the receiving area is used to update the order to reflect the receipt. This module will decrement quantity-left-to-receive and automatically close the order, if appropriate. It will also increment on-hand inventory or inspection inventory as required. In addition, a hard-copy material receiving document can be printed to accompany the parts for accounts payable control. If backorders exist for a part being received, a document can be generated indicating the quantity of parts backordered and the department requiring them. This insures prompt filling of backorders once an out-of-stock part is received.

Material is issued from stock to work orders, backorders, or extra usage requirements. Normally, stock will be issued against a hard-copy picking list, which contains all the pull requests for a particular work order on a particular date. Just prior to the start date of a work order, picking lists for each inventory location are produced. After the material has been pulled, a terminal is used to enter the work order number and a picking list identification number. With just this one entry, all necessary issue transactions for that picking list will be processed. Each complete issue will automatically delete demand, update the inventory counts, and create an audit trail record. A partial issue will automatically generate a backorder. Backorders are filled automatically in user-specified priority sequence as soon as the backorder item is received into stock.

INVENTORY BALANCE MANAGEMENT

Inventory Balance Management maintains the current inventory status of every part in the stockroom. Inventory status information includes quantity on hand, quantity in inspection, and quantity available for release. This module is also responsible for the maintenance of the physical warehouse locations where inventory is kept.

Inventory Balance Management Features

- On-line inventory balance update and retrieval.
- Multiple stocking locations for each part.
- Inventory balance by location.
- Two stage cycle counting based on ABC code.
- Inspection inventory.

Using Inventory Balance Management

Receipt, issue, and stock adjustment transactions will trigger an immediate update of the inventory counts as well as create an audit trail record. Up to date, on-hand status of every part can be reviewed on-line. A cycle counting system based on total usage value of the parts (ABC value classification) helps ensure inventory accuracy.

The multiple stocking location feature allows for the processing of inventory control transactions by inventory location. A virtually unlimited number of warehouses and locations within warehouses can be specified for each part.

WORK ORDER CONTROL

The Work Order Control module maintains the information required to track scheduled receipts (open work orders) and to monitor scheduled issues (both allocations and backorders) required for work order assembly.

Work Order Control Features

- On-line work order entry, update, and retrieval.
- On-line allocation of parts to work orders.
- On-line review of allocations by part number or work order number.
- Variable allocation window by part.
- Automatic management of backorders.
- Acceptance of extra usage requirements.
- Shortage/preshortage report.
- Work order tracking.

Using Work Order Control

A work order to a production facility is entered at a terminal by the production control department, and includes the quantity of a particular part required by a specified due date. This work order; since it is an authorization to build an assembly, part, or product; requires the issue of on-hand inventory from stock for its completion. These requirements are calculated by exploding the bill of material for the ordered part to create allocations which, on the appropriate date, will become requisitions for the correct amount of each component. The resulting set of scheduled issues may be reviewed from a terminal, compared with all other competing allocations for the same part, and modified manually if required. If a particular warehouse is specified on the work order, all component issues will be made from that warehouse, if unreleased inventory is available.

By having allocations available before the actual issue of parts from inventory; preshortage reports, which match all allocations for a particular part to the quantity available, are produced to point out possible parts shortages.

PURCHASE ORDER TRACKING

The Purchase Order Tracking module maintains information required to track the scheduled receipt of materials and parts purchased from vendors. Descriptive information about each vendor as well as information on purchase commitments is also maintained.

Purchase Order Tracking Features

- On-line purchase order entry, update, and retrieval.
- Purchase commitment report.
- Purchase order tracking.
- Vendor name and address file.
- Purchase order receipt offset.

Using Purchase Order Tracking

A purchase order to be issued to a vendor is entered at a terminal in the purchasing department, and includes necessary tracking information such as quantities and due dates, as well as descriptive data. Purchase orders may then be reviewed by part number, line item, or inventory controller. The receiving of purchase orders is provided for in the Material Issues and Receipts module. Because this tracking information is maintained, a purchase commitment report can be produced specifying weekly cash requirements to meet the purchasing plan.

MATERIAL REQUIREMENTS PLANNING

Material Requirements Planning is a priority planning technique used in a manufacturing operation to help optimize materials control. Utilizing information from the other Materials Management/3000 modules, Material Requirements Planning determines the timing and quantities of parts to be purchased or produced and identifies the current orders to be rescheduled. Material Requirements Planning is a regenerative system; it recalculates the entire materials plan on each cycle. The timing of this replanning can be adapted to any desired cycle.

Material Requirements Planning Features

- Six standard order policies:
  - —Fixed order quantity.
  - —Lot for lot.
  - —Part period balancing.
  - —Days of supply.
  - —Order Point.
  - —Gross requirements.
- Rescheduling of work orders and purchase orders.
- Five year, bucketless planning horizon.
- Firm planned order.
- Single-level pegging of requirements by work order number and part number.
- Automatic recalculation of ABC codes.
- Standard hard-copy reports for order planning, reschedules, and new suggested orders.
- No action report.
- Summarized inventory planner report showing inventory position, inventory requirements, and suggested actions.

Using Material Requirements Planning

Material Requirements Planning takes the master schedule, bills of material information, and current inventory and order status information to produce a series of reports used by production control and purchasing to plan inventory procurements.

Once all demands for a part or assembly have been determined, Material Requirements Planning will allocate current inventory and orders to those demands and then suggest new orders based on the part's order planning algorithm. The current and suggested orders are then offset by the assembly lead time, modified by yield factors, and exploded via the bill of material to form dependent demand for each assembly's component parts.

The parts controller (inventory planner) receives an exception report that highlights all Material Requirements Planning suggestions for "push outs" and "pull ins" of existing orders, as well as any new suggested orders that should be examined before the next Material Requirements Planning run. Suggested orders are made according to individual controller parameters.

Controllers may take appropriate action based directly on the exception report, or may prefer to investigate the situation that triggered the suggested action. An action report, which displays all supply and demand entries for each part, can be consulted to determine the cause of the exception. The parts controller has the ability to override any Material Requirements Planning actions. Once decisions have been made, the controller may take action to reflect the plan for order reschedules and new order releases.

STANDARD PRODUCT COSTING

Standard Product Costing performs the calculations necessary to determine the current total cost for each subassembly, assembly, and end item in a manufacturer's material planning and control system. It can also set standard costs for each product and its associated components. Standard costs are carefully determined costs and may be used for setting goals, establishing budgets, measuring performance, or determining product prices.

Standard Product Costing Features

- Systematic cost editing and Cost Roll-Up Edit Report.
- Cost edit and roll-up by effectivity date.
- Current cost roll-up without affecting established standard costs.
- Current to standard cost roll-over.
- Parts selection for roll-up and roll-over by any one of four methods:
  —Part number.
  —Product line.
  —Account number.
  —Workcenter.
- Optional level-by-level roll-up.
- Product Cost Sheet for each part costed.
- Costed Bill of Material.

Using Standard Product Costing

Standard Product Costing uses customer supplied current cost information normally entered by purchasing personnel for material costs, production control personnel for workcenter and routing information, and accounting personnel for labor and overhead rates.

Cost editing identifies current costs that have not been properly initialized or have logical inconsistencies, such as a purchased part with no material cost. A Cost Roll-up Edit Report is printed that identifies all detected conditions that could cause an unreliable or inconsistent roll-up.

Cost roll-up is a major function of Standard Product Costing. Roll-up may be stopped at the completion of each assembly level, or it may be done non-stop through all levels. Specific part selection, using one of four different criteria, provides a high degree of flexibility. Costs at each level are accumulated and then combined with the costs for all lower levels before being rolled up to the next higher level. The roll-up continues until the current cost is determined for the selected part and is accomplished without affecting the standard cost for the selected part. A product cost report is printed for each selected part and each of its fabricated components. These reports can then be used by the finance or accounting department to determine the acceptability of the calculated costs for their use as standard costs.

Cost roll-over converts the results of the latest cost roll-up to standard costs by replacing the old standard cost values with the existing current cost values. Cost roll-over can also be done on a specific part selection basis. Standards are established only after all information is edited, accumulated, and verified.

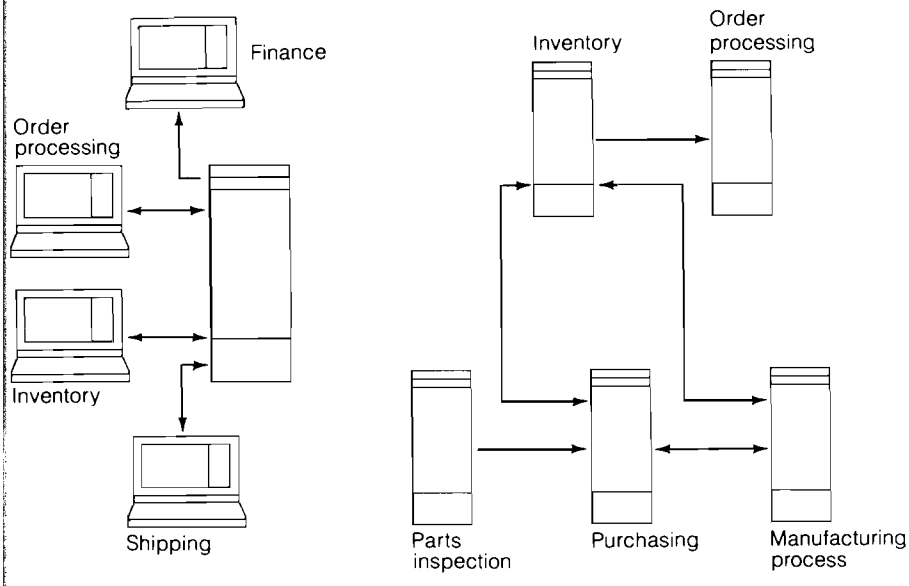
Chapter 4

DATA COMMUNICATIONS

HEWLETT-PACKARD DISTRIBUTED SYSTEMS NETWORK (HP-DSN)
TERMINAL COMMUNICATIONS
DISTRIBUTED SYSTEMS/3000
DS/3000: A MANAGEMENT PERSPECTIVE
COMMUNICATIONS WITH IBM MAINFRAME COMPUTER SYSTEMS
HP-IBM COMMUNICATIONS: A MANAGEMENT PERSPECTIVE
IML/3000
MRJE/3000
RJE/3000
MODEMS

HEWLETT-PACKARD DISTRIBUTED SYSTEMS NETWORK (HP-DSN)

Hewlett-Packard Distributed Systems Network (HP-DSN) is the high-level, user-oriented network architecture that allows HP 3000 computer systems to communicate in distributed data processing networks. As shown in Figure 4-1, HP-DSN allows processing to be distributed either geographically or functionally in your organization. However, it is flexible enough to allow for either centralized or decentralized control. HP-DSN networks need not adhere to any particular configuration; hierarchical, linear, or circular networks of HP 3000 nodes in any combination are accommodated by HP-DSN. There is no absolute limit to the number of HP computer systems that may communicate in an HP-DSN network. HP-DSN allows communication between the HP 3000 and other HP computer systems, such as the real-time, interrupt-driven, HP 1000. Further, batch and interactive communication between HP 3000's and IBM or IBM-compatible mainframe systems are also included in HP-DSN (Fig. 4-2). These capabilities allow you to choose the HP system which exactly meets your requirements—be it for general accounting or factory data collection—and be assured of compatible communications between systems. The IBM communication capabilities allow you to optimize the trade-off between totally centralized and totally decentralized processing, and provide a smooth transition from strictly centralized to distributed processing.



If you need more timely information or want to improve information accuracy in your manufacturing operations or if you need to integrate a comprehensive manufacturing information network into the operational aspects of managing your business, or if you want to exchange information interactively among a number of manufacturing sites... with HP-DSN you can define and implement a distributed processing system to handle your current requirements with the appropriate level of control while providing a growth path to meet your future needs.

Figure 4-1. HP's range of distributed processing solutions lets you put computer power where the work is done.

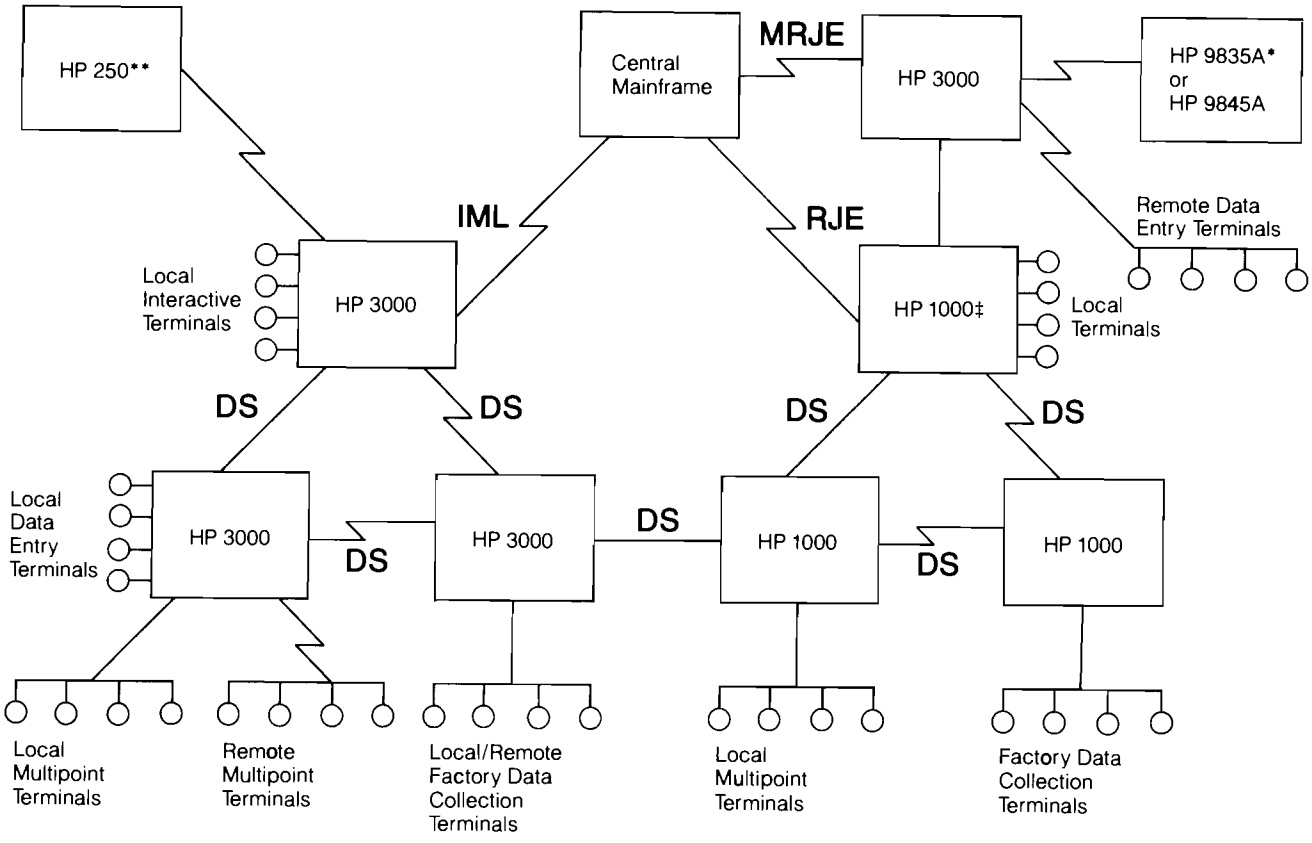


Figure 4-2. Hewlett-Packard Distributed Systems Network (HP-DSN)

* HP 9835A and 9845A are desk-top computers that can act as terminals to HP computer systems.

** The HP 250 system can act as a terminal to the HP 3000.

‡ The HP 1000 is a scientific, real-time, interrupt-driven computer system, widely used in factory data collection and instrumentation environments. Devices that are compatible with IEEE Std. 488-1975 (HP-IB) are supported by the HP 1000.

HP-DSN Objectives and Implementation

HP-DSN consists of a set of design objectives, and a set of hardware and software products that implement these objectives. The objectives of HP-DSN may be summarized as follows:

- high-level user interfaces (user freedom from having to know communications protocols or line characteristics)
- device and network independence
- sharing of resources among systems (hardware, software, and data)
- communication with both HP and non-HP equipment
- network diagnosis and recovery
- data and communication integrity and security

The Hewlett-Packard Distributed Systems Network is implemented as a number of functional layers in software with only the high level system services visible to you. The layered approach, depicted in Figure 4-3 provides flexibility for additions and enhancements at each layer. Advantages of this approach include both stability and flexibility: stability because changes can be made to the internal layers without requiring you to alter application programs, and flexibility because the modular design of the network easily accommodates changes as a result of technological improvements. The specific functions assigned to each layer are described below.

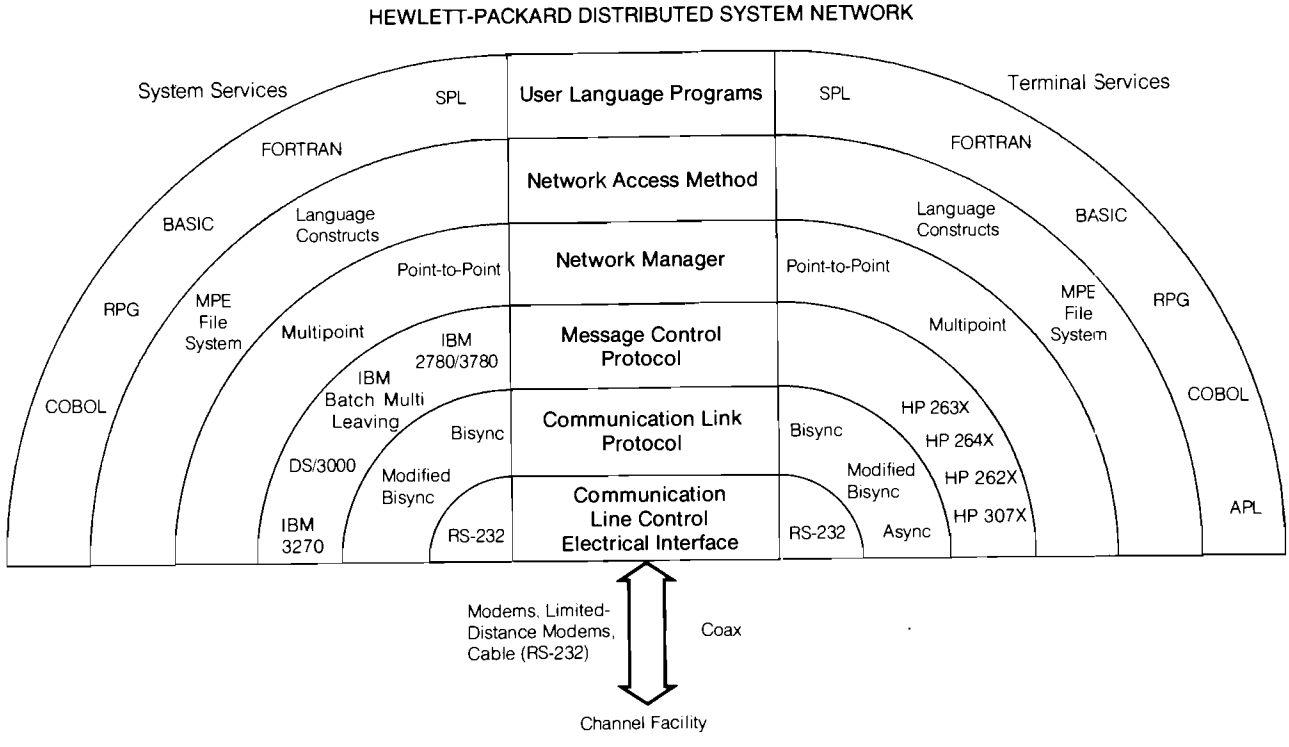


Figure 4-3. HP DSN is implemented as a series of software layers. Improvements can be made in any layer without affecting the highest level, your application programs.

Network Access Method

The network access method gives you the network capabilities at the application level through callable MPE intrinsics, system services, and specific language I/O constructs. The services include:

- Remote file access
- Remote data base access
- Remote command execution (virtual terminal access)
- Remote peripheral access
- Remote program management

Network Manager

The network manager layer is aware of the network topology and is responsible for managing network error recovery and the various topology dependent functions such as polling.

Message Control Protocol

This layer provides control functions, addressing information, message type, and other requirements to effect end-to-end transmission.

Communication Line Protocol

The communication line protocol layer takes care of the "grammar" or protocol by which two or more systems can exchange information in an efficient and reliable manner.

Communication Line Controller

This is the physical interface that defines the number of circuits and the meaning of the signals on those circuits.

HP-DSN Products

HP-DSN products fall into two categories: software and hardware. HP-DSN software subsystems, all of which run under the control of the MPE operating system, are:

- MTS/3000—Multipoint Terminal Software/3000—synchronous, multipoint or multidrop terminal communications (asynchronous terminal communications are a capability inherent in MPE and do not require an additional HP-DSN software subsystem).
- DS/3000—Distributed Systems/3000—interactive data communications between HP computer systems
- IML/3000—Interactive Mainframe Link/3000 (IBM 3270 Emulation) high-level, interactive data communications between the HP 3000 and IBM mainframe.
- MRJE/3000—Multileaving Remote Job Entry/3000—emulation of a multileaving batch workstation (IBM 360/30), providing job management services for HASP II, JES 2, JES 3, and ASP job entry subsystems mainframe
- RJE/3000—Remote Job Entry/3000
  —IBM 2780/3780 emulation

HP-DSN hardware includes:

- Synchronous Communications Interfaces—
- Synchronous Single Line Controller (SSLC)
- Hardwired Serial Interface (HSI)
- Intelligent Network Processor (INP)

Modems—

- 4800 bits/second, dial up or leased lines
- 9600 bits/second, leased lines

and the Asynchronous Repeater for multipoint communication. The Reference Sheets in the HP 3000 Specification Guide should be consulted for further information on the hardware products.

The HP-DSN software and hardware products afford a complete solution to your distributed processing requirements from Hewlett-Packard—except for the telephone line!

TERMINAL COMMUNICATIONS

The terminal handling capabilities of the HP 3000 provide you with a number of alternatives for use in designing the most cost-effective and functional communication links to terminals. Asynchronous or synchronous communication may take place over point-to-point or multipoint links. Links may use modems, or be hardwired, with several transmission speeds possible.

Point To Point Terminal Communications

Point to point terminal communications is provided on the HP 3000 Series III by the Asynchronous Terminal Controller (ATC). Up to 4 ATC's can be connected to give a maximum of 64 terminals including the system console. Modems supported by the ATC are Bell types 103A, 202T, 212A and Vadic VA3400.

For the Series 30 and 33, the Asynchronous Data Communication Controller (ADCC) allows 4 terminals to be connected. Each ADCC main can have one ADCC extender to allow a total of 8 terminals. The maximum number of terminals is 32 for the Series 33 and the Series 30. Modems supported by the ADCC are Bell types 103A, 212A, and 202T.

The maximum speed for an asynchronous terminal on an ATC is 2400 bps and 9600 bps on an ADCC. For dial-up connection, speed sensing is automatic up to 2400 bps. Terminals can be configured to the operating system (MPE) as data entry terminals under your program control, or as log-on terminals accessing all the capabilities of the HP 3000. Terminals normally operate in character mode, except when accessed via HP V/3000 when block mode is employed. If you wish to access terminals in block mode directly (i.e., not using V/3000), you must provide the detection and correction facilities for transmission errors by calling operating system routines.

Multipoint Terminal Communications

Multipoint Terminal Software (MTS/3000) permits 9600 bps data transmission between the HP 3000 and multiple HP 2640 series multipoint terminals using a single communication line. The terminals may be connected to the computer by means of a modem (remote access) or may be hardwired to the computer (local access). With MTS/3000 you use MPE commands and file system intrinsics to communicate with the terminals. You may also initiate sessions from the terminals and thus access all the resources of the MPE operating system.

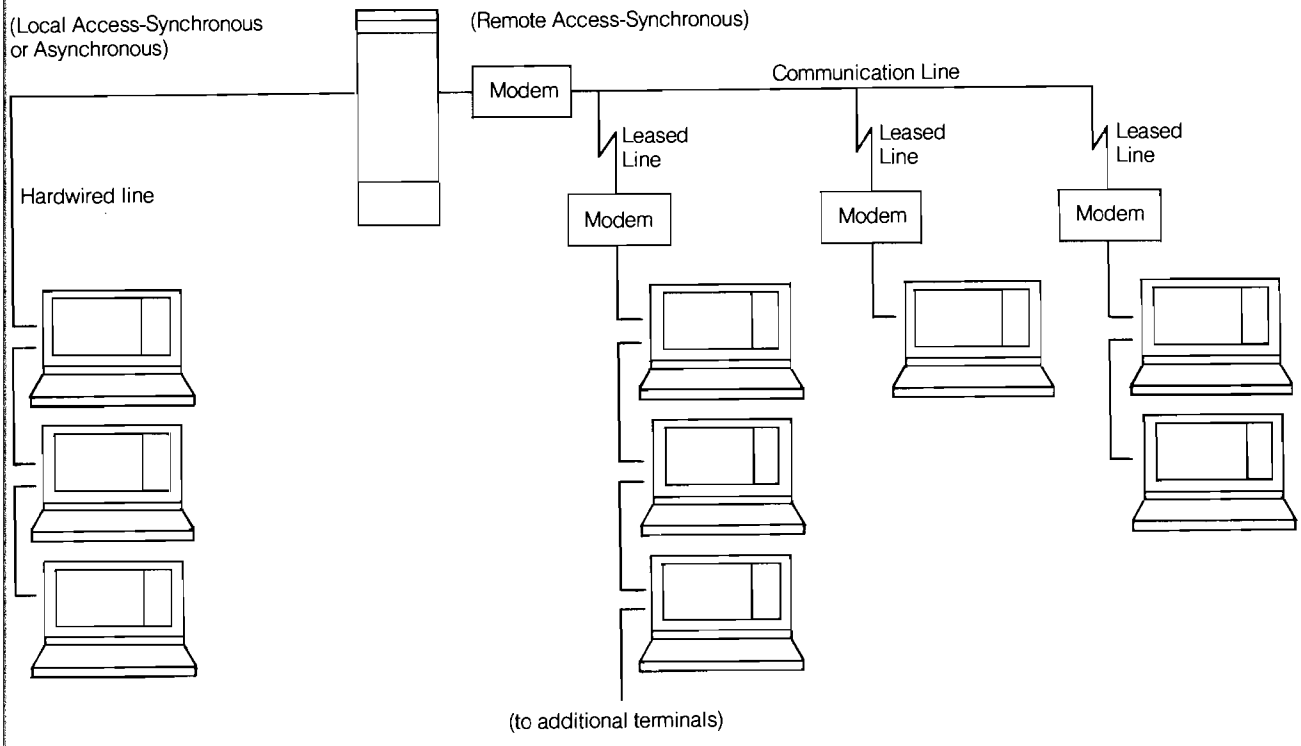


Figure 4-4. Typical MTS/3000 network

Terminal Networks

Figure 4-4 is an illustration of a network of multipoint terminals. When terminals are connected by modems (remote access) communication is synchronous; hardwired terminals (local access) may be either synchronous or asynchronous. Up to 2000 feet (610 meters) of cable may separate individual hardwired terminals. However, the first terminal must be within 50 feet (15 meters) of the system unless using the Asynchronous Repeater to extend the cable distance.

Data Transmission

Terminals may transfer data at speeds up to 9600 bps. They may operate in either log-on (interactive) or data entry mode. In either mode, you enter data into the terminal's memory using the cursor positioning capabilities, TAB key, and RETURN key. This data can then be edited as much as desired until the ENTER key is pressed, transferring the data to the computer. Variable length blocks of data may be transferred in this manner.

Power Down Bypass Cable

A power down bypass cable is available for use with MTS/3000. The cable, ordered with the multipoint terminal, enables the system to bypass a terminal which has no power, so that the remaining terminals in the daisy chain group are not affected.

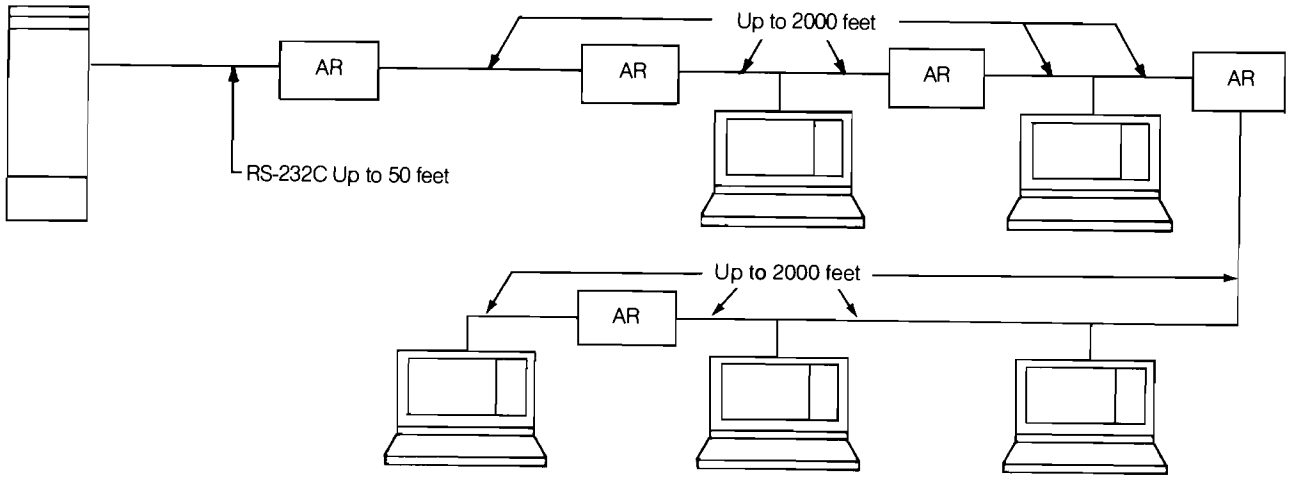


Figure 4-5. Use of the asynchronous repeater in multipoint terminal environment

Longer Distance Direct Connections

The Asynchronous Repeater (AR) is a stand-alone device which converts standard RS232C communication signals to levels compatible with the HP 2640 series of terminals. It is useful in either an asynchronous multipoint or point-to-point environment.

By adding an AR in a multipoint environment, the first directly connected terminal on a line can be located up to 2000 feet (610 meters) from the computer. This removes the 50 foot (15 meter) limitation imposed when an RS232C direct connect interface is used. In addition, each AR can extend the maximum cable distance between individual multipoint terminals (or a group drop of terminals) by an additional 2000 feet (610 meters). Figure 4-5 shows examples of the possible AR uses in a multipoint terminal environment. Note that additional ARs may be added to further extend the maximum distance allowed.

ARs may also be used in the point-to-point environment where terminals are attached to the system via the Asynchronous Terminal Controller.

Terminal Peripheral Devices

The multipoint terminal can be viewed as a processor with several peripheral devices:

- keyboard
- display
- two cartridge tape units (optional)
- printer (optional)

These peripheral devices can be controlled by transmitting the appropriate escape sequences to the terminal from your application program. The capabilities provided by the escape sequences include data transfers from the HP 3000 to a peripheral device, from a peripheral device to the HP 3000, and from one peripheral device to another within the same terminal. An escape sequence also exists to retrieve status information for a peripheral device.

These capabilities can be used with both point-to-point and multipoint communications. In multipoint transmission the peripherals share the terminal's multidrop line. In a point-to-point environment using the Asynchronous Terminal Controller (ATC), the peripheral shares the point-to-point line.

DISTRIBUTED SYSTEMS/3000

The Distributed Systems/3000 (DS/3000) software subsystem is used for intercommunication between HP computer systems. It provides a complete set of network communications services so that programs, files, peripheral devices, and processing can be shared in a network. No knowledge of communications programming is required to use DS/3000. In fact, most DS/3000 capabilities are available to you with no programming effort once the software and required hardware interfaces are installed.

DS/3000 allows an HP 3000 to communicate with both HP 3000 and HP 1000 systems. The HP 1000, with its Real-Time Executive (RTE) operating system, is a high-performance computer system designed for real-time computation and instrumentation applications.

Communication between systems occurs in a bidirectional interleaved fashion using hardwired coaxial cables for HP 3000 to HP 1000 communications, and both hardwired and modem connections for linking HP 3000 systems.

DS/3000: A Management Perspective

DS/3000 was designed to be particularly useful in commercial applications that involve transaction processing and are geographically dispersed. DS/3000 has four advantages of special interest to management:

- Easy user access among all HP 3000 computers in the network
- Simplified network management
- Ease of network implementation
- More efficient use of all systems resources

Easy User Access

In an HP Distributed Systems Network, any HP 3000 user has every other HP 3000 in the network at his fingertips. All system commands may be executed remotely simply by inserting the word REMOTE in front of the command. No knowledge of the communication link used —be it a direct connection (hardwired link) or telephone link—is required. Programmers can access remote data files using the normal input/output statements of each programming language, just as if the files were local. No special training for users or programmers is required.

Simplified Network Management

DS/3000 makes the system or network manager's job easier. Although computer power is distributed to the location where it is needed, control of the network may be maintained centrally. The remote command processing feature allows a single system manager to control all satellite HP 3000 systems from the central EDP facility, if desired. The manager may assign each user different capabilities on different computers in the network with each operating system automatically enforcing the capability assignments. This affords functional dedication of each computer in the network. For example, all program development may be done on one machine, all batch on another, and all on-line transaction processing on a third computer.

Changes in network topography can be made with no reprogramming. For example, when a company with two divisions using a single HP 3000 grows to the size that each division needs its own computer system, the two systems can be linked together and access common files with no application program changes. A network manager may define and re-adjust his network to best meet the needs and growth of his organization, without having to worry about the effects on existing programs and files.

Data security and accounting for resource usage are responsibilities of the network manager. DS/3000 automatically affords full data security for each computer in the network. Full control of which users can read or write specific data is provided. Given the proper security codes, either user or manager can obtain exclusive access to any specified link in the network. Resource usage is automatically logged for CPU time, disc space, and connect time for all users, whether remote or local.

Ease of Implementation

Using DS/3000, the network may be configured in any combination of rings, stars, or strings—whichever best reflects the structure of your organization. Among HP 3000's, you have a choice of switched, leased, or hardwired lines. These different line types may be mixed throughout the network, so you can choose the most convenient and/or economical type of line for each particular link.

Efficient Use of System Resources

With DS/3000 several systems can share the use of expensive peripherals via communications lines, thus keeping capital investment to a minimum without sacrificing capability. The peripherals appear to you just as if they were on your local system. Each physical communications line in an HP Distributed Systems Network can be used concurrently by many different batch and interactive applications originating from either end of the line. DS/3000 automatically processes all data being transferred over each line, so that each application is essentially unaware that anyone else is using the same line.

Time is a precious commodity in any organization. Within an HP Distributed Systems Network, a wide variety of processing can be in progress simultaneously, including local and remote batch jobs, problem solving, inter-system program-to-program communication, and remote job entry to your mainframe.

3000/3000 and 3000/1000 Communications Links

The HP 3000 to HP 3000 and HP 3000 to HP 1000 communications links give you the opportunity to select the appropriate blend of computing power for your specific requirements. After either of these communications links has been established, two commands enable you to make use of DS/3000 network services to other network nodes. The command DSLINE identifies the remote computer. Telephone number, identification list, maximum transmission buffer size, and the activation of data compression may also be specified. The other command, REMOTE, is used to direct locally entered commands to a remote computer identified in the DSLINE command. With these commands, the following capabilities are available:

- Remote Command Execution (RCE) allows users of a local system to employ exactly the same set of operating system commands as are available to a user at the remote system.
- Remote file access (RFA) gives you full access to the data files on a remote system.
- Remote peripheral access (RPA) works just like RFA, giving you full access to peripheral devices on the remote computer.
- Remote data base access (RDBA) lets you access a remote data base on a remote computer system on a transaction-by-transaction basis.
- Program-to-program communications allows your application programs being run on separate systems to interactively exchange data and control information.

The following paragraphs contain additional information about each of these capabilities.

Remote Command Execution

With remote command execution you can execute the entire set of MPE commands on a remote HP 3000 while connected to a local HP 3000 or a local HP 1000. You can also execute system level RTE operator commands on a remote HP 1000 while connected to a local HP 3000.

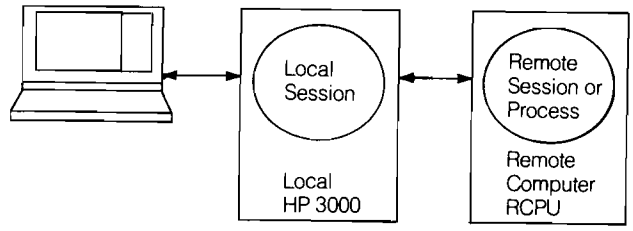


Figure 4-6. Remote command execution allows you to issue commands to a remote system as if the local terminal were connected directly to the remote system.

To execute a remote MPE or system level RTE operator command from a local HP 3000, you simply enter the REMOTE command at the local terminal as follows:

:REMOTE command

where command is the desired MPE or RTE operator command in its normal format.

The following is an example of HP 3000 to HP 3000 communications. Commands are issued from a local terminal to a remote computer for execution.

```
:HELLO USER.ACCT ———————— Log on to local HP 3000
:DSLINE RCPU ———————— Designates the remote system for remote processing

:REMOTE
#HELLO RUSER.RACCT ———————— Log on to the remote system and entry of commands to be executed on the remote computer
#LISTF
#EDITOR
#SHOWJOB
    .
    .
    .
#BYE ———————— Log off the remote HP 3000
```

Issuable HP 3000 to HP 3000 commands include all user commands, system supervisor commands, account manager commands, and system manager commands.

You can use the same REMOTE command to issue standard HP 1000 system level commands to be executed on a remote HP 1000 computer system.

The procedure for issuing a remote MPE command from a local HP 1000 has a slightly different format. After the remote HP 3000 is accessed and the communications link is established, you need enter only the following at the local HP 1000:

#command

where command is the desired MPE command in its normal format.

Replies generated at the remote HP 3000 are returned to you at your terminal on the local HP 1000.

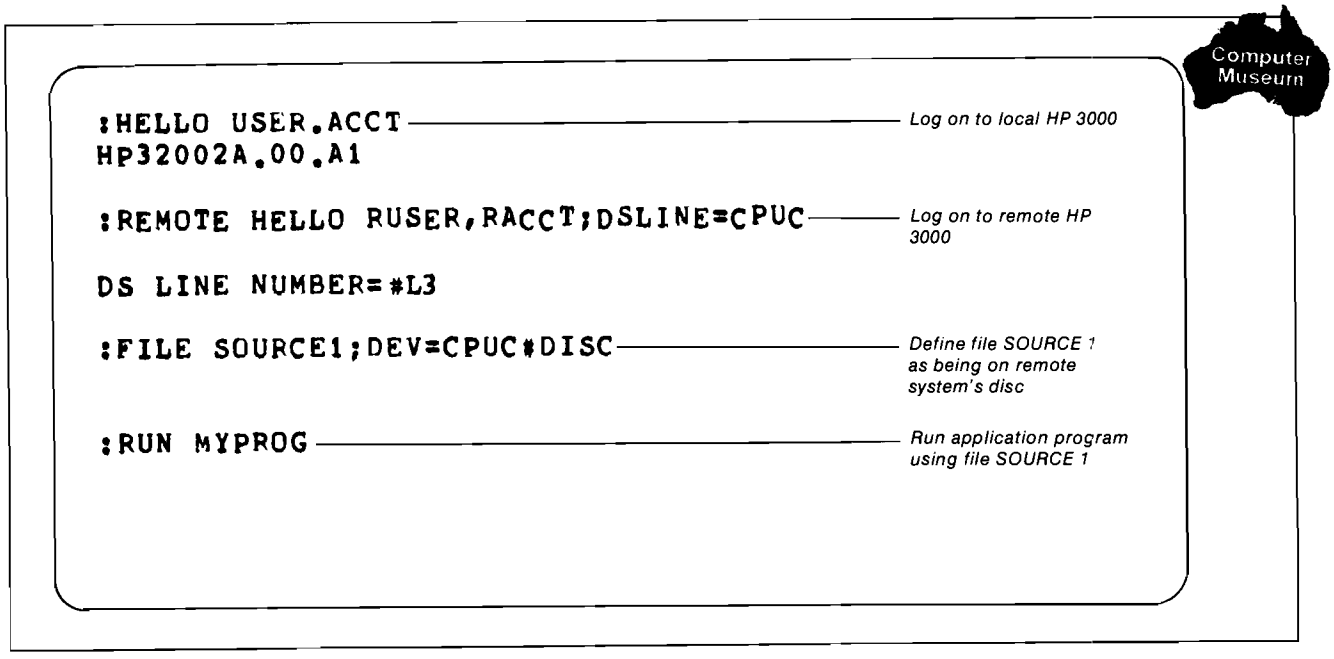
Remote File Access (RFA)

With DS/3000 linking an HP 3000 with either another HP 3000 or an HP 1000, you have access to the files of the remote system. There are three methods by which you can access remote files: utility programs (3000/3000 link only), standard language input/output statements (3000/3000 link only), and remote file access intrinsics (3000/1000 link).

HP 3000 to HP 3000 RFA: With the utility programs and editor, you merely issue a local MPE FILE command on your terminal prior to running the utility. The FILE command is used to define the desired remote file. Included in the FILE command must be a remote device specification denoting the location of the desired file. The utility is run as though it were accessing a local file.

Access using standard language input or output statements permits local programs written in any language to define files and manipulate complete files or file records on another remote HP 3000. All that is required is a FILE command which may be made external to the program or may be included in the program. This command specifies the location of the target file. Subsequently, the remote file may then be utilized on a record-by-record basis or as a complete file as though it resided on your local computer.

For example, you can run your local application program (MYPROG) which accesses the remote file SOURCE1 with the following command sequence:



Local HP 3000 application programs may utilize the remote file access intrinsics to access standard MPE files which reside on a remote HP 3000. In addition to accessing standard MPE files on a remote HP 3000, DS/3000 also allows you to access remote KSAM/3000 (Keyed Sequential Access Method) files using standard KSAM intrinsics from your local HP 3000.

Regardless of which method is used, the system file security available on a single HP 3000, is maintained across the DS/3000 link.

HP 3000 to HP 1000 RFA: To access files on an HP 1000's moving-head disc, flexible disc, or tape mini-cartridge from an HP 3000, you can write an HP 3000 program that makes use of a set of intrinsics, called remote FMP, that allow access to the remote HP 1000 file system. These intrinsics have a direct correspondence to the standard HP 1000 RTE file management package (FMP).

HP 1000 to HP 3000 RFA: Local HP 1000 application programs may use the remote file access intrinsics to define, control, and access disc files on a remote HP 3000.

To access HP 1000 files, you enter the following at the local terminal:

Remote Peripheral Access

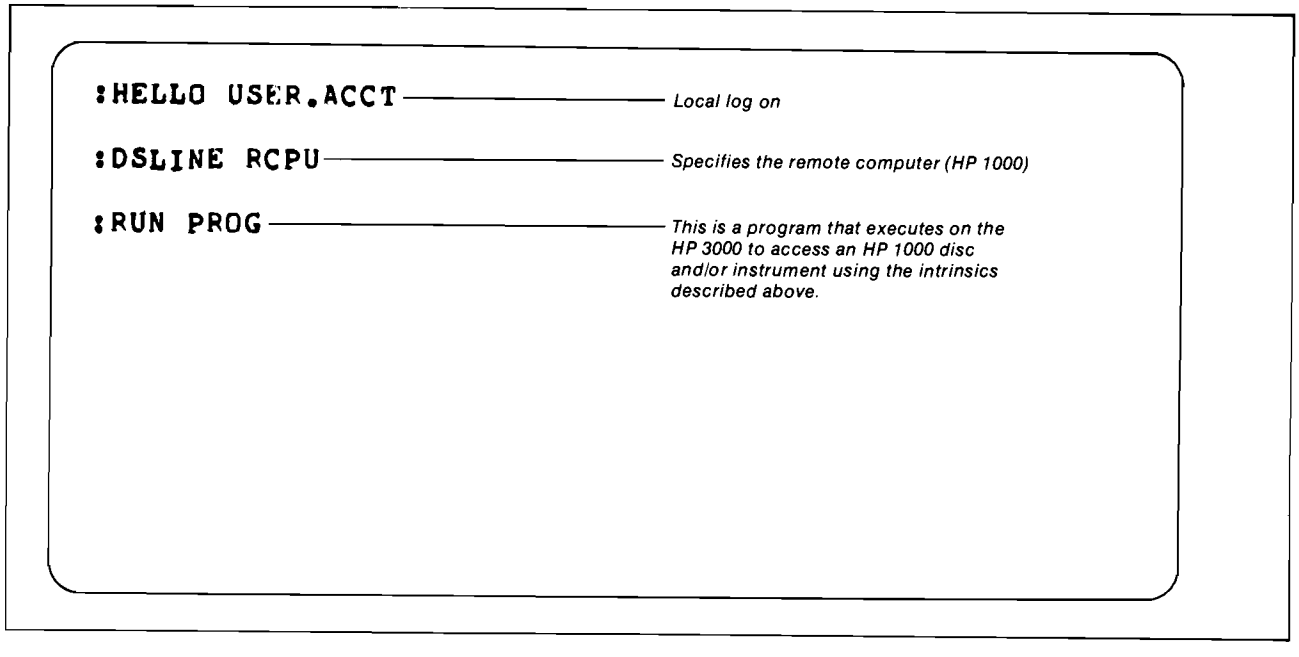
Accessing remote peripheral devices can be exactly the same as accessing remote files. The methods which exist for accessing remote files: HP 3000 utility programs and the standard language input/output statements, are available.

In addition, HP 1000 peripheral devices, such as line printers, magnetic tape units, and scientific instrumentation, can be accessed by an HP 3000 program using remote Distributed Executive (DEXEC) calls.

These DEXEC calls may also be used to schedule or terminate programs, request system time, or inquire about the status of a program or an I/O device. These intrinsics have a direct correspondence to standard RTE EXEC calls.

Remote Data Base Access

The remote data base access feature of DS/3000 gives you the capability of direct and indirect access of data bases on remote computer systems. Using the QUERY and IMAGE data base inquiry facilities of the remote system you can locate, report, and update data values directly in remote IMAGE/3000 data bases and indirectly in remote IMAGE/1000 data bases.



Remote Data Base Access (RDBA): The remote data base access feature of DS/3000 gives you the capability of direct and indirect access of data bases on remote computer systems. Using the IMAGE and QUERY data base inquiry facilities of the remote system you can locate, report, and update data values directly in remote IMAGE/3000 data bases, a remote program is not required.

Direct data base access is a feature of the HP 3000 to HP 3000 communications link only. Using this method, you can automatically interrogate data bases on remote HP 3000 computer systems with remote QUERY interactive commands. Your local application programs can also contain standard IMAGE calls which retrieve and manipulate information in the remote data base. Intervention from a remote program is not required.

There are three methods by which you can directly access a remote data base. The first method requires you to establish a communications line and a remote session and enter a FILE equation for each remote data base. The FILE equation specifies which data base is to be accessed on which remote system and device. A local IMAGE application program can then be run to access the remote data base. The second method allows you to embed MPE command calls within your program, removing the need for the user to know that the data base is remote. The third method requires that a file called the data base access file (DBA file) be created. This file provides IMAGE with the necessary information to establish a communications link and a remote session. It also specifies the remote data base file name so that the necessary IMAGE intrinsics can be executed on the remote computer. The following example illustrates this method. The DBA file is built by the HP 3000 EDITOR. It is named DBAX and contains:

```
REC1 FILE DBX;DEV=RCPU# DISC
REC2 DSLINE RCPU
REC3 USER.ACCT,GROUP=HELLO RUSER.ACCT
  .
  .
  .
RECN
```

Record 1 specifies the location of the remote data base

Record 2 specifies the remote computer on which data base resides.

Records 3 through n specify which local user, group and account may access which user, group, and account on the remote computer.

As a local HP 3000 user, you simply type the following to access the remote data base DBX:

```
:HELLO USER.ACCT ———————————— Local log on

:RUN PROGX ———————————— PROGX is an IMAGE program
                         that opens DBAX,
                         automatically establishes
                         communication with the
                         remote HP 3000, and accesses
                         the remote data base.
```

Indirect data base access is available on both the 3000/3000 and the 3000/1000 communications links, using the program-to-program capability. By first initiating a program on your local system, you can then run a remote program which accesses and/or updates the remote data base. The remote program then passes the requested information back to your local program.

Program-to-Program Communications (PTOPC): The DS/3000 program-to-program communications capability gives you the ability to write application programs using a set of nine intrinsics. The two programs can be in different languages. These intrinsics make it possible for two or more user programs residing on different computer systems to exchange data and control information directly and efficiently with one another. The intrinsics, called PTOPC intrinsics, are directly callable by SPL, FORTRAN, BASIC, and COBOL programs. The nature of any two programs communicating with one another in this manner is not symmetrical. One of them (referred to as the master program) is always in control and is the one that initiates all activity between the two programs. The other (referred to as the slave program) always responds to requests received from the master (see Figure 4-7). The master program opens the data link, initiates the slave program, and is always in control. The slave program merely responds to requests received from the master program, either to accept or reject the master program requests. Each computer may have master and slave programs active simultaneously, depending on the needs of your specific application.

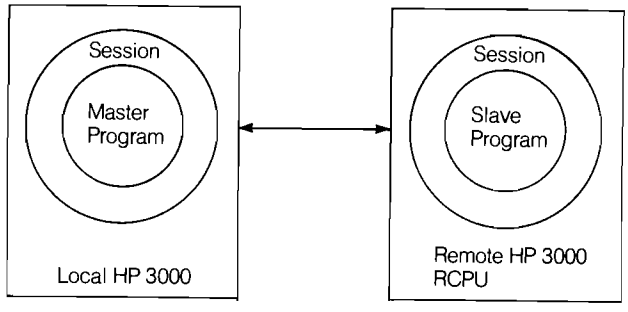


Figure 4-7. Program-to-program communication allows user programs on different systems to execute and exchange data in a coordinated manner. One of the programs is the master and is always in control. The slave program responds to the master's requests.

COMMUNICATIONS WITH IBM MAINFRAME COMPUTER SYSTEMS

Whether you require a single remote job entry link, or a world-wide batch/interactive network of several hundred HP 3000's and IBM computers, Hewlett-Packard's HP-IBM communications for the HP 3000 provide the solution to your current and future requirements. The HP 3000's advanced networking software for communication with IBM systems (or IBM plug-compatible systems such as Amdahl, Itel, or Magnuson) is designed to be easy to use, to maximize programmer productivity, and to allow sharing of peripherals among HP and IBM systems. Hewlett-Packard's HP-IBM communications for the HP 3000 consists of three software subsystems:

- Interactive Mainframe Link/3000 (3270 Emulator)
- Multileaving Remote Job Entry/3000 (HASPII/JES/ASP Workstation Emulator)
- Remote Job Entry/3000 (2780/3780 Emulator)

Interactive Mainframe Link/3000 (IML/3000) is the HP 3000's sophisticated 3270 emulator software. It provides high-level, user-oriented, interactive communication between IBM mainframes and application programs or terminal users on the HP 3000. IML/3000's Programmatic Access and Update allows user-written application programs on the HP 3000 in COBOL, BASIC, FORTRAN, or SPL to exchange data interactively with CICS, IMS DB/DC, TSO, or other mainframe application programs. Designed for maximum HP 3000 programmer productivity, IML/3000's Programmatic Access and Update handles low-level, detailed programming work, allowing more time to be spent on application design and less on time-consuming coding tasks. IML/3000's Inquiry and Development facility (IDF) allows, with a few keystrokes, an HP terminal to be switched between the HP 3000 and access to the mainframe as an IBM 3270 terminal. IML/3000 uses HP's Intelligent Network Processor (INP) to off-load much of the overhead of 3270 communications from the HP 3000 CPU.

Multileaving Remote Job Entry/3000 (MRJE/3000) software allows multiple jobs to be transmitted and received from one or more hosts at the same time. When used with HASP II, JES 2, JES 3, or ASP job entry subsystems on the host, MRJE/3000 allows job output to be automatically routed to any desired files or output devices on the HP 3000. Simple user commands make it easy to control job input or output from any terminal or batch job running on the HP 3000. Remote Job Entry/3000 (RJE/3000) may be used to submit and receive batch jobs from the HP 3000 as an IBM 2780 or 3780 batch terminal, or it may also be used to communicate (pass and receive files) with many non-IBM systems that emulate IBM 2780/3780 devices.

Together, the three products allow you to design distributed processing networks consisting of any number of HP 3000's and/or mainframe computers. All three software products run under the control of the HP 3000's Multi-Programming Executive (MPE-III) operating system, and may be used concurrently with each other or any other HP 3000 processing.

HP-IBM COMMUNICATIONS: A MANAGEMENT PERSPECTIVE

Batch Payroll or On-Line Inventory

HP-IBM communications software span a broad range of applications in a distributed processing environment. RJE/3000 and MRJE/3000 can be used where batch transmissions of data from the HP 3000 to the mainframe are required, such as payroll processing or quarterly reporting. RJE/3000 may also be used for batch input or output to interactive applications (such as CICS) on the mainframe. For transaction-oriented applications such as on-line updating of inventory data bases, IML/3000 is the solution. All three products—RJE, MRJE, and IML—can, of course, be used at the same time on the same HP 3000.

Friendliness and Ease of Use

Hewlett-Packard's HP-IBM communications software is designed to minimize required training and familiarity. MRJE commands, for example, are English words such as HOST, SUBMIT, and EXIT that are easy to remember. For applications using IML/3000 intrinsics (high-level system routines), all user programming is done in high-level languages; IML/3000 takes care of all low-level manipulation of screen control characters. IML/3000's inquiry and Development Facility (IDF) allows terminals to appear to the user to be switched from the HP 3000 to the host with the simple keyboard command "IML." For unskilled users of IDF, the "Auto Acquire" feature allows the IML Manager to make HP terminals enter IDF automatically—with no user intervention required.

Central Control—If Desired

With the HP 3000's remote console capability, HP-IBM communications can be managed from the mainframe site, if desired. A single HP 3000 system console at the mainframe site can dial-up different remote HP 3000's to run IML/3000 programs or transmit batch jobs through MRJE/3000 or RJE/3000, allowing unattended operation for many applications.

Efficient Use of System Resources

No special peripherals are required for input and output with HP-IBM communications products. RJE/3000 and MRJE/3000 allow jobs to be submitted and output to be received through any ordinary input or output device or file on the HP 3000. Any HP terminal—not just the system console—can be used by the MRJE manager to control workstation activity, or by the IML manager to control programmatic and IDF use of IML/3000. And IML/3000's Inquiry and Development Facility (IDF) allows HP terminals to be shared between the mainframe and the HP 3000.

A Smooth Transition to Distributed Processing

The HP 3000's HP-IBM communications can let your EDP operation make a smooth transition from centralized to distributed processing. Existing host applications that support 3270 terminals can usually be used with IML/3000 with little or no modification. Alternatively, by developing new host applications for use in conjunction with HP 3000 IML programs, you can optimize the transfer of information between the two computers, and off-load teleprocessing function from your host mainframe. Or, new on-line applications can be implemented directly on the HP 3000, while still having access to information stored on the host. By providing a gradual transition to distributed processing, IML/3000 minimizes the impact on your EDP operation and budget.

INTERACTIVE MAINFRAME LINK/3000 (3270 Emulator)

IML/3000 allows the HP 3000 to appear to the IBM host as a bisync (BSC), IBM 3271, 3274, or 3276 terminal controller ("cluster controller"). Interactive communication can take place in either of two ways: through high-level, user-written application programs running on the HP 3000 (Programmatic Access and Update), and through HP 264x block-mode terminals on the HP 3000 (Inquiry and Development Facility). HP programs and terminals act as 3277 or 3278 connected to the host system (Fig 4-8). The HP 3000 can also share multidropped leased lines with the 3270 terminals you already have (Fig 4-9).

High-Level Application Programming

The programmatic Mode of IML/3000 allows HP 3000 programs in COBOL, FORTRAN, BASIC and SPL to read data from, and pass data to, teleprocessing applications on the host system. HP 3000 programs communicate through IML/3000 "intrinsics" (high level system routines) with standard versions of IBM access methods (BTAM, TCAM, or VTAM), and standard communication monitors that support IBM 3270 terminals, such as CICS, IMS DB/DC, or TSO. Program development with IML/3000 intrinsics is strictly in high-level languages: all low-level decoding of 3270 screen-format commands is done automatically by IML/3000.

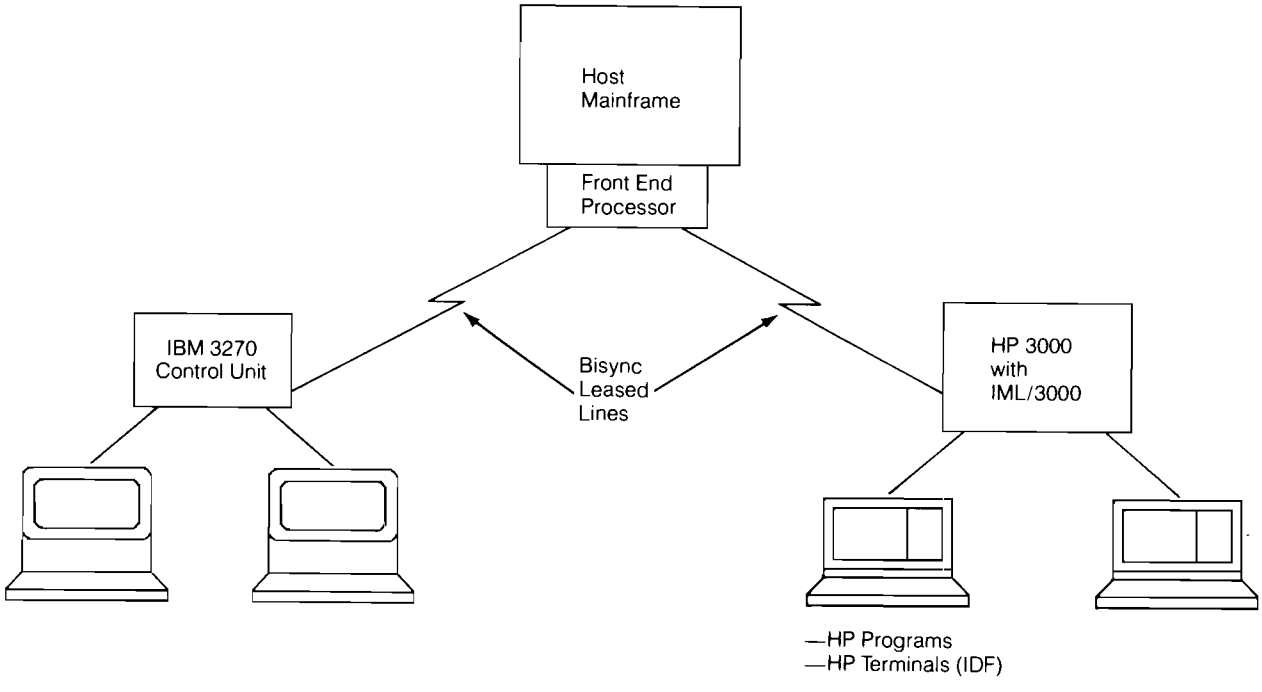


Fig 4-8. Interactive Mainframe Link/3000 (3270 Emulator.) To the host mainframe, IML/3000 makes the HP 3000 appear as a remotely-connected IBM 3270 control unit with attached terminals.

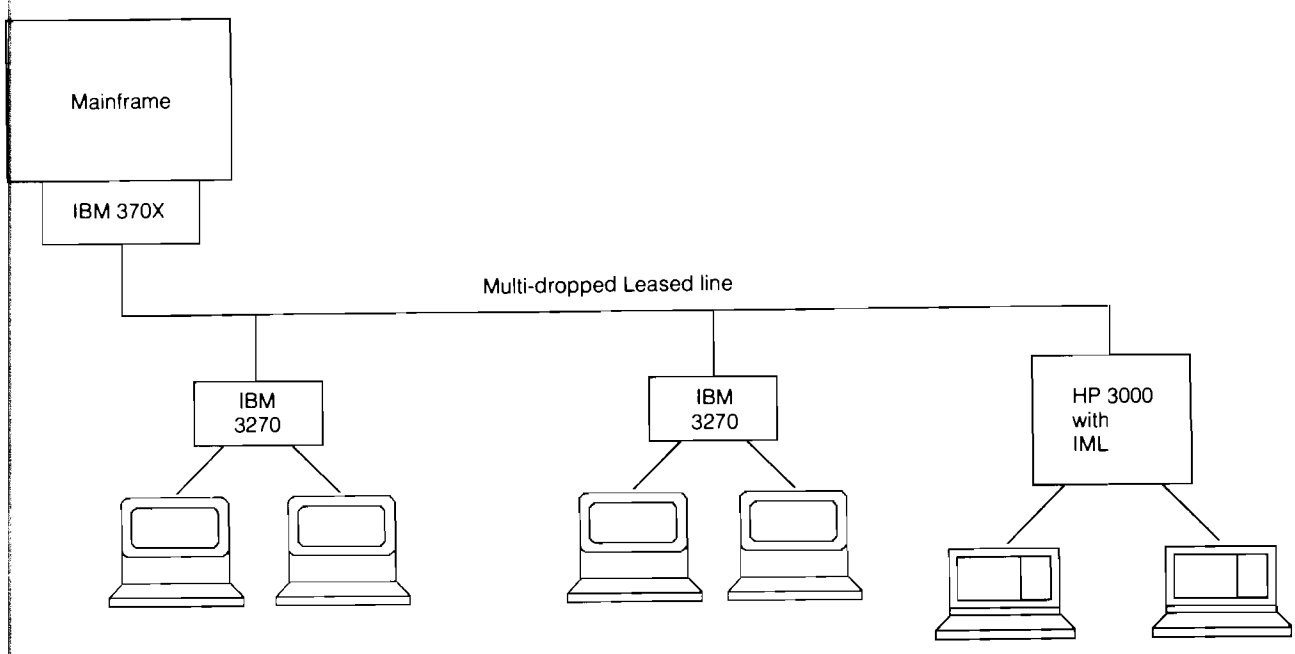


Fig 4-9. Leased lines can be shared. IML/3000 allows one or more HP 3000's to be multi-dropped on the same leased lines that support 3270 terminals.

Transparent, Programmatic Access of Mainframe and HP Data Bases

HP 3000 programs containing IML/3000 intrinsics may use the intrinsics of any other HP 3000 software product to let you construct very powerful and flexible applications. For example, a single HP 3000 program may contain, in addition to IML/3000 intrinsics, V/3000 (Hewlett-Packard's interactive data entry/retrieval software for the HP 3000) intrinsics, and IMAGE/3000 (Hewlett-Packard's Data Base Management System for the HP 3000) intrinsics. V/3000 intrinsics provide screen-formatting and editing for on-line data entry from HP terminals; IMAGE and IML intrinsics can allow data being entered to be transmitted to either the local HP 3000 or mainframe databases, respectively. No knowledge of the ultimate destination (HP 3000 or mainframe data base) of input data—or source of accessed data—is required by the terminal user. Thus, the HP 3000 and mainframe data bases can be made to appear to the terminal user as a single, integrated data base (Fig 4-10). By including DS/3000 (Distributed Systems/3000) intrinsics in the HP 3000 application as well, data bases on any number of remote HP 3000's can be made to appear similarly integrated (Fig 4-11.)

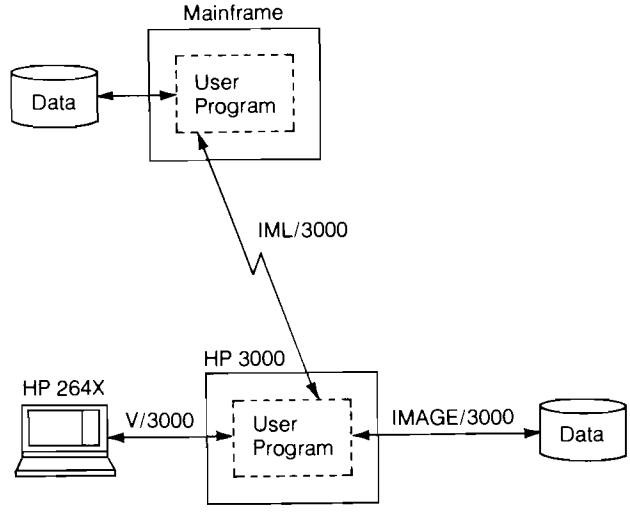


Fig 4-10. IML/3000 Programmatic Access and Update. By using IML/3000 in an HP 3000 program with V/3000 and IMAGE/3000, separate data bases on the mainframe and HP 3000 can be made to appear to HP terminal users as a single, integrated data base.

IDF: No Programming Required

The IML Inquiry and Development Facility (IDF) allows HP terminals attached to the HP 3000 to emulate principal features of IBM 3277 or 3278 terminals attached to an IBM 3271, 3274, or 3276 control unit. A few keystrokes on the HP terminal keyboard are all that is required for the terminal to be switched between a local HP 3000 session and direct access through IDF to a remote host mainframe. Inquiry and Development Facility may be used either with teleprocessing applications such as CICS or IMS DB/DC programs, or for access to timesharing services such as TSO. Inquiry and Development Facility also serves as an aid in developing and testing HP 3000 applications which use the IML program statements. No programming on the HP 3000 is required to use the Inquiry and Development Facility.

Most block-mode members of the HP 264x terminal family may use the Inquiry and Development Facility. Functions of the IBM 3277 and 3278 terminals emulated by HP terminals using IDF include Program Function (PF) keys and Program Attention (PA) keys. The function keys on HP 264x terminals are used to implement certain features of the IBM 3270 terminals: All Program Function (PF) keys, PA1, PA2, PA3, CLEAR, and EXIT. Users should expect that response times of HP terminals using IDF will be longer than those associated with IBM 3270 terminals. Consult the IML/3000 Reference Manual for a detailed description of the differences between an HP 264x using IDF and an IBM 3277 or 3278 terminal.

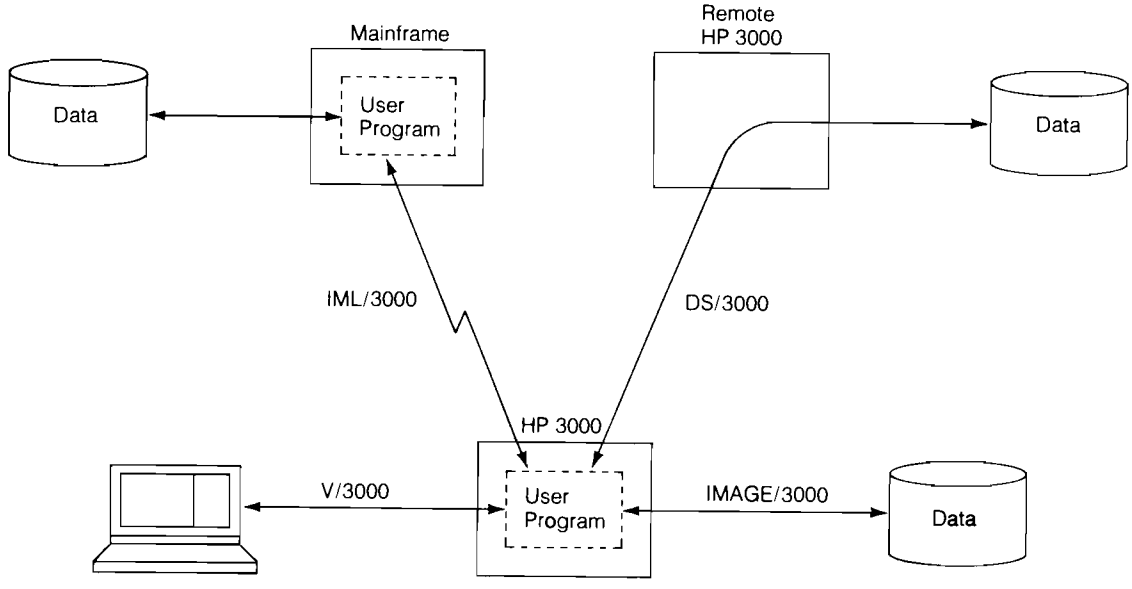


Fig 4-11. IML/3000 and DS/3000. Together, IML and DS can allow any number of HP 3000 and mainframe systems to communicate interactively.

IML Configuration File: Security and Flexibility

IML/3000 allows a person (or persons) designated as IML manager to manage, through an HP 3000-resident configuration file, which IML users can use Inquiry and Development Facility (IDF) for each terminal on the HP 3000, and which HP 3000 programs can act as IML devices. This affords a high degree of security for mainframe files or data bases from unwarranted access through the HP 3000. The configuration file further provides flexibility for the IML manager in assigning HP terminals, programs or printers ("HP devices") as logical devices in the emulated 3270 control unit. No physical reconfiguring is ever necessary to assign different HP devices to different 3270 logical devices. The IML "Auto Acquire" feature allows the IML manager from the HP system console to cause HP terminals to enter Inquiry and Development Facility (IDF) automatically—simplifying use of IDF for untrained users.

Intelligent Network Processor

IML/3000 uses the Intelligent Network Processor (INP) to off-load the overhead of the 3270 control unit processing from the HP 3000 CPU. IML/3000 software is down-loaded into the INP from the HP 3000 CPU at the time that the IML/3000 subsystem is initiated. The same INP which is used for IML/3000 may be used at different times for other HP 3000 communications subsystems as well. (Only one subsystem may be down-loaded into the INP at a time.) For IML/3000, the INP supports line speeds up to 9600 bits per second point-to-point, and 7200 bits per second when the HP 3000 is multi-dropped on the same communication line with other IBM 3270 control units or HP 3000's using IML/3000.

To use IML/3000 to communicate with more than one host at a time, one INP is required per host for each HP 3000. Only one copy of the IML/3000 software is required per HP 3000, regardless of the number of different host computers with which the HP 3000 is communicating.

IML/3000 Training/Consulting Package

The IML/3000 Training/Consulting Package consists of that combination of training and consulting required to enable your programmers to bring up new HP 3000-to-host applications as quickly as possible. The package consists of a total of four days on-site customer training and implementation consulting, conducted after IML/3000 installation on your site by an HP Systems Engineer. The package covers design and development of IML/3000 user applications, management of the IML/3000 subsystem, implementing security, and use of IDF. The package is recommended for every initial installation of IML/3000.

MULTILEAVING REMOTE JOB ENTRY/3000

MRJE/3000 gives multiple users on the HP 3000 simultaneous batch access to any remotely connected host computer system using a HASP II, JES 2, JES 3, or ASP job entry system. Job data may be submitted from the HP 3000 via any ordinary input devices (disc files, magnetic tapes, card readers, or terminals) and output received to the HP 3000 via any output devices (printers, disc files, tapes, card punches, or terminals). MRJE/3000 jobs may be submitted, and job input and output devices on the HP 3000 specified, by either an interactive session at an HP 3000, or a batch job running on the HP 3000. In addition to providing for multiple MRJE/3000 users, MRJE/3000 provides for an MRJE Manager who can interactively monitor and control job activity. Host console commands can be entered by the MRJE manager from any HP terminal.

Features

- May be used with HASP II, JES 2, JES 3, or ASP job entry subsystems on the host computer
- Flexible, easy-to-use commands for job submission and status inquiry
- Any input/output devices on the HP 3000 may be used to submit or receive jobs (disc file, magnetic tape, card reader, card punch, printer, or terminal)
- Job submission capability available to multiple users simultaneously
- Supports for concurrent use: an operator console, up to seven logical print streams, seven logical card reader streams, and seven logical punch streams, all interleaved on the same communication line
- Supports multiple hosts and/or multiple lines to a single host
- May use either switched (dial-up) or leased lines; modem communication speeds up to 9600 bits per second
- Jobs may be submitted, and job input/output devices on the HP 3000 specified, by either an interactive session on an HP terminal, or by a batch job running on the HP 3000

Submit Jobs On-line or Off-line

MRJE users may submit MRJE jobs even if no connection exists between the HP 3000 and the mainframe. Jobs submitted off-line are spooled on the HP 3000 and automatically transmitted when the connection is made. Output from the host is then directed to the proper HP 3000 peripheral device or file without further intervention. If no output destination has been indicated, job output is routed to a default device designated by the MRJE manager. MRJE/3000 output may be sent directly to a printer without intermediate spooling, while simultaneously spooling local HP output for the printer, if desired.

MRJE/3000 is designed to be used in a full multiprogramming environment (i.e. it may be used concurrently with any other HP 3000 processing.) However, the HP 3000 with MRJE/3000 is not recommended as a dedicated batch workstation for high-volume remote job entry to the mainframe. Your local Hewlett-Packard Systems Engineer can assist you in evaluating your batch workstation application, in assessing its impact, if any, on local HP 3000 processing, and in planning MRJE activity to maximize overall system performance. A minimum memory size of 512 Kbytes for the HP 3000 is often recommended for MRJE/3000.

The hardware interface between the HP 3000 and the communication line for MRJE/3000 is the Synchronous Single Line Controller (SSLC).

REMOTE JOB ENTRY/3000 (HP 2780/3780 EMULATOR)

RJE/3000 makes the HP 3000 appear to the host system as either an IBM 2780 or 3780 data transmission terminal, and thus allows batch jobs for the host system to be submitted and received from the HP 3000. RJE/3000 provides greater flexibility than the IBM data communication terminals it emulates, in that jobs may be submitted from the HP 3000 via any ordinary input device (disc file, magnetic tape, card reader, or terminal) and output received to the HP 3000 via any output device (printer, disc file, tape, card punch, or terminal). RJE/3000 jobs can be transmitted to the host, and job input and output devices on the HP 3000 specified, by either an interactive session from an HP terminal or a batch job running on the HP 3000.

RJE/3000—Where To Use It

RJE/3000 can be used for remote job entry to any host computer which supports the IBM 2780 or 3780 communication terminals. In addition to use with mainframe job entry subsystems, RJE/3000 may be used with mainframe communication monitors such as CICS for batch input/output to interactive applications. It can also be used to communicate (transmit and receive files) with the IBM 2780 or 3780 themselves, or other devices or systems that emulate them. RJE/3000 is often used as a means for communicating with other vendors' minicomputers.

Features

- Any input/output device on the HP 3000 may be used to submit or receive jobs (disc file, magnetic tape, card reader, card punch, printer, or terminal)
- Supports lines to multiple hosts and/or multiple lines to a single host
- Can be used with either switched (dial-up) or leased lines; Modem communications up to 19.2 Kilobits per second with the Intelligent Network Processor (INP) hardware interface for the HP 3000.
- Jobs may be submitted, and input/output devices specified, by either an interactive session on an HP terminal, or by an HP 3000 batch job.

In addition to the capabilities of the IBM 2780 or 3780, RJE/3000 provides the following advantages:

- When emulating an IBM 2780, RJE/3000 performs short-record truncation (suppression of trailing blanks) without the user having to supply EM (End of Medium) control characters in the data.
- Unlike an actual IBM 2780 which cannot do character compression, RJE/3000 can compress blank fields when emulating an IBM 2780.
- When emulating an IBM 2780, RJE/3000 can block more than seven records.

As an interface between the HP 3000 and the communication line, RJE/3000 can use either the Synchronous Single Line Controller (SSLC) or the Intelligent Network Processor (INP). One SSLC or INP interface is required per concurrent user of RJE/3000. ASCII and EBCDIC character codes are supported by RJE/3000. IBM 2780 six-bit transcode is not supported.

MODEMS

Hewlett-Packard synchronous modems can be used with DS/3000, RJE/3000, MRJE/3000, MTS/3000 and IML/3000. They are fully compatible with the HP 3000 and can be used with the Synchronous Single Line Controller (SSLC) or Intelligent Network Processor (INP). A wide variety of options is available, including comprehensive self test facilities.

Two modems are available, one operating at 4800 bps and the other at 9600 bps. The 4800 bps unit will operate on switched or leased lines and can be used for multidrop configurations. The 9600 bps unit is intended for point-to-point applications using leased lines. The units allow a wide variety of configurations and offer considerable flexibility when data networks are being constructed.

Chapter 5

THE OPERATING SYSTEM–MPE III

EFFICIENT VERSATILITY
A USER-ORIENTED OPERATING SYSTEM
USER INTERFACE
PROGRAM DEVELOPMENT
A DYNAMIC ENVIRONMENT
SYSTEM OPERATION

The functional heart of the HP 3000 is the Multiprogramming Executive, MPE III. This general purpose, disc-based operating system supervises all processing and maintains all user interface with the HP 3000. Two major attributes of MPE are its versatility and ease of use.

Designed to take full advantage of the computer's hardware features such as virtual memory and stack architecture, MPE demonstrates its versatility by enabling the HP 3000 to perform transaction processing, on-line program development, data communications, and batch processing concurrently. In addition, MPE permits system resources to be accessed simultaneously by multiple users, each of whom interfaces with the system independently.

MPE demonstrates its ease of use with its many user assistance features such as a powerful, straight-forward command language and an on-line HELP facility which guides you in using MPE commands. In addition, MPE simplifies the programming task by monitoring and controlling program input, compilation, execution, and output. MPE regulates the order in which programs are executed, and dynamically allocates any hardware and software resources the programs require.

A complete account structure and automatic resource accounting are standard features of MPE. Easy to use MPE commands allow the system manager to set up a hierarchical accounting structure on the system in a style similar to a company organization chart. MPE then automatically keeps track of the system resources used by the various groups in the account structure. This resource usage information can then be used for billing, accounting, or any other application that requires such data.

MPE provides complete security, enabling you to operate in an environment protected from interference or illegal access by other users. This security is accomplished by means of multiple logon passwords, file lockwords, hierarchical access restrictions, and user capability sets.

MPE handles all input/output to peripheral devices, receiving the I/O requests, queueing them if necessary, and performing the actual data transfer. Because MPE treats I/O devices as files, you can write programs without concern for the physical source or destination of the data, and you can run them in either batch or interactive mode without changing the names of the files they reference.

MPE also handles asynchronous terminal communications. Synchronous data communication to terminals, other HP computers, or even non-HP computers is provided by optional data communications subsystems described in Chapter 4.

Figure 5-1 illustrates the major components of the Multiprogramming Executive III Operating System.

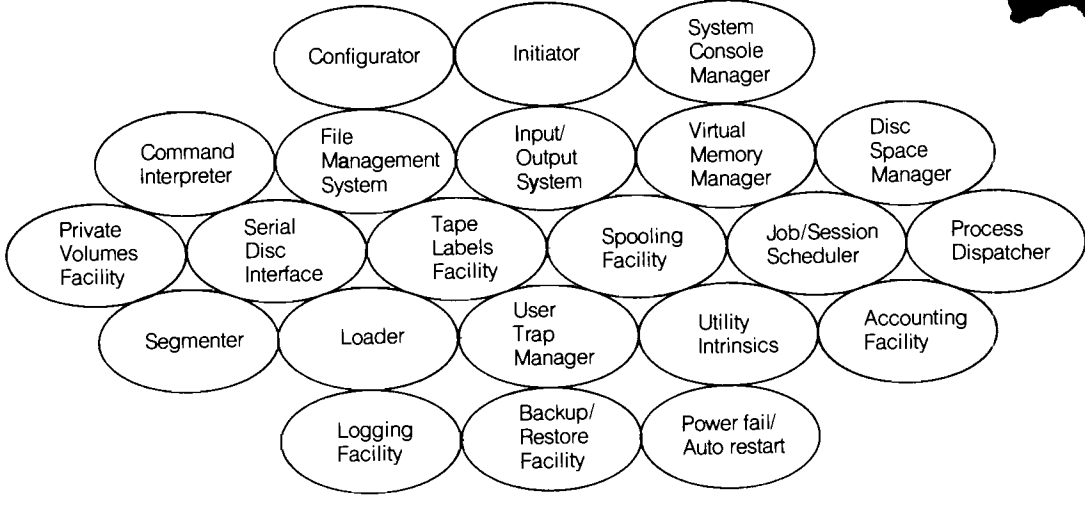


Figure 5-1. Components of the MPE operating system.

EFFICIENT VERSATILITY

All HP 3000 computer systems operate under a single operating system—MPE. This means that programs prepared under MPE III can be run without modification on any other HP 3000 operating under MPE III, provided that all devices required by the programs are connected on-line. It also means that you can move from one HP 3000 to another without undergoing additional training for a new environment.

Multiprogramming

One of the major ways in which operating efficiency is achieved in an HP 3000 is by multiprogramming—the concurrent execution of multiple programs. Multiprogramming allows system resources to be allocated among several competing programs. While one program is awaiting an I/O operation, for instance, control of the central processor is shifted to the next highest priority program waiting for the CPU. MPE is designed to allocate, schedule, and dispatch control of the central processor, storage, and input/output devices among the competing programs. This controlled competition for system resources reduces turnaround time, and significantly increases system throughput. Operating in conjunction with the architecture of the central processor, MPE provides complete protection against one program interfering with another.

The number of programs that can be processed concurrently depends upon such factors as hardware configuration, program operating modes, and the application programs involved. MPE is designed so that the maximum number of concurrently running programs can be increased or decreased by changing a single system configuration parameter.

MPE allows the concurrent execution of programs from two types of input media—traditional batch input devices and interactive terminals. Programs are independent of their input mode and the same system code is used to perform particular functions in either mode. This results in storage economy and reduced overhead.

Interactive Processing

When using the interactive processing mode, you enter commands and data through a keyboard terminal and receive immediate responses to your input. This type of interaction is called a session and is especially useful for program development, text editing, data entry, information retrieval, computer-assisted instruction, and other applications where a direct dialogue with the system is preferred. Sessions can be used to access:

- Operating system commands and subsystems
- Language and utility programs
- Data base management programs
- Data communications programs
- Application programs

A session begins when you enter the :HELLO command from an on-line terminal and MPE connects you to the command interpreter. You may then enter commands to use language compilers or other subsystems such as the text editor, to run programs, or to modify your files. The session continues until you enter a :BYE command, a new :HELLO command, or the system operator intervenes to abort the session.

As an example, let's assume you want to create a COBOL program and then compile, prepare, and run it during an interactive session. Figure 5-2 shows the various commands entered during such a session. You initiate the session by pressing the RETURN key on a terminal that is connected on-line to the system. MPE responds by displaying a colon prompt character. You log on to MPE by entering a HELLO command containing your assigned user and account names, then call the HP 3000 text editor and enter two editor commands followed by the COBOL statements that constitute your program. When the entire source program has been entered, you save it on disc under a file named YOURFILE by entering a KEEP editor command, and then terminate the editor subsystem. Now the source program exists as a disc file in the system. To compile, prepare, and execute the program, you simply enter a COBOLGO command specifying the file YOURFILE. This one command first invokes the COBOL compiler which compiles the program, then invokes the MPE segmenter which prepares the compiled program into an executable form, and finally executes the prepared program. When the program has finished executing, MPE displays the message END OF PROGRAM followed by another colon prompt character. You terminate the session by entering the BYE command.

The example below is somewhat simplified since it does not include the various informational messages, compilation output, and program output generated by MPE, the text editor, the COBOL compiler, and the program itself. The fact remains, however, that if the source program (entered by way of the editor) contains no errors, the entire session can be performed by entering just-eight MPE and text editor commands in addition to the COBOL statements which constitute the program.

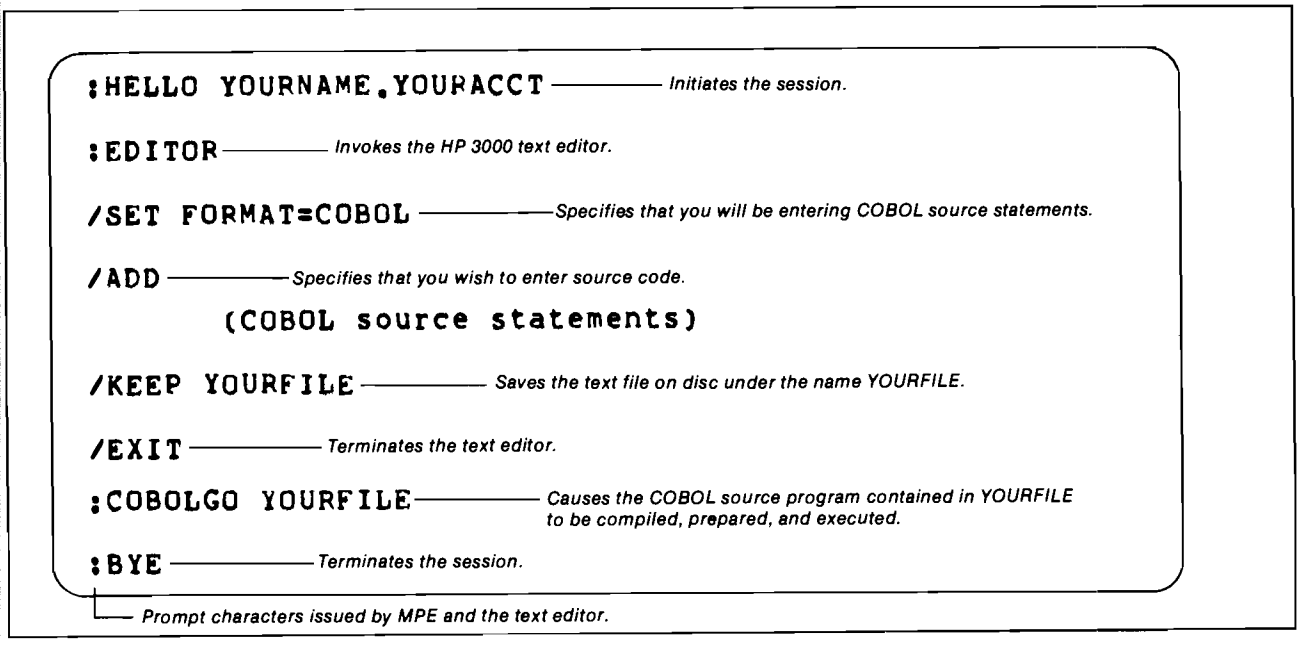


Figure 5-2. Sample session

Batch Processing

Batch processing is a logical extension of the interactive functions available through MPE. Any capability, with the exception of BREAK, that is available in one mode, is available in the other and employs the same MPE commands. Languages, utilities, and applications development software can be run in either batch or interactive mode without changes. The standard input and output devices are automatically redefined.

The batch processing mode lets you submit to the computer, as a single unit, commands that request various MPE operations such as program compilation and execution, file manipulation, or utility functions. Such a unit is called a job. Jobs contain all necessary instructions to MPE and all references to programs and data required for their execution. Once a job is running, you need to supply no further information.

Jobs are often read through batch input devices such as card readers or tape units. A unique feature of MPE, however, allows you to enter batch job streams through the terminal during the course of an interactive session.

Several jobs can be submitted to the system from multiple devices concurrently. Batch job input is spooled on disc and MPE schedules each job for execution according to its job input priority specified in the JOB command. Additional commands are provided for monitoring job selection. The system operator specifies the maximum number of jobs that can be executed concurrently and can dynamically adjust the job selection criteria.

When a job enters execution, the commands within it are executed sequentially on a multiprogramming basis. MPE 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 job or session (if available) immediately enters execution. Spooled output on disc is selected for output processing according to the output priority specified in the JOB command.

MPE executes many sessions and batch jobs simultaneously. The only significant difference between a session and a batch job is that during a session you can interactively alter the course of processing, whereas in a job the the command stream is fixed and the job will be executed in its entirety, as pre-defined in the job control statements, without active intervention.

A USER-ORIENTED OPERATING SYSTEM

The many features and capabilities of the Multiprogramming Executive operating system are designed around the concept of the user. There are two distinct types of users:

- the end user, such as a data entry clerk, whose only concern is running a compiled application program, and
- system administrators, such as application programmers, who are responsible for the creation and maintenance of application programs as well as the day-to-day operation of the system.

Each type of user is associated with a particular set of capabilities and responsibilities, and each has access to MPE features which assist him with his specific tasks.

User Classification

Programmers are users who create application programs which run on the system. MPE provides two major areas of system interface for these users: an interactive interface which includes a command language, an on-line HELP facility, and job control facilities; and a programmatic interface which includes programming intrinsics and the MPE file system.

The end user, who can range from an order entry clerk to a functional manager, takes advantage of all the capabilities of the operating system through an application program which he can run without any knowledge of MPE itself.

System administration is performed by a hierarchy of users whose overall responsibility is the successful administration of the computer system. The various levels of MPE administration are defined in Table 5-1. In a small installation, a single user may perform the functions associated with all levels of administration. In a larger installation, the capabilities may be divided among several individuals at each administrative level. Thus it is more appropriate to think of "a user with system manager capabilities" rather than a formally titled "system manager."

TABLE 5.1 SYSTEM ADMINISTRATORS

System Manager:
Manages the overall system by creating accounts (basic structures for user access) and defining resource-use limits.

System Supervisor:
Manages the system on a day-to-day basis, controls scheduling queues, alters system configuration, and maintains the system library.

Account Managers:
Maintain accounts by defining the valid users and file groups for the accounts and specifying resource-use limits for them.

The system operator is the user who operates the system console and is responsible for responding to all system requests. MPE provides a range of operational capabilities which augment the performance of day-to-day operations such as system start-up, back-up, maintenance and recovery, as well as helping the system operator keep the system operating as smoothly and efficiently as possible.

User Capabilities

Capability sets are assigned to each system user based on the kinds of tasks each user needs to perform. These capability sets are divided into three categories:

- User attributes
- File access attributes
- Capability-class attributes

User attributes include system manager, system supervisor, and account manager capabilities. Additional capabilities such as account librarian with special file access capabilities for maintenance of account files, group librarian with special file access capabilities for maintenance of group files, and diagnostician with the ability to run diagnostic programs under MPE for on-line checkout of HP 3000 hardware components may also be assigned.

There are two file access attributes: *save*, which permits the user to save files by declaring them permanent, and *non-shareable devices*, which allows a user to use non-shareable devices, such as a magnetic tape unit.

Capability-class attributes refer to the ability to access special MPE facilities. Included here are interactive access, local batch access, process handling, extra data segment acquisition, multiple resource identification numbers, privileged mode, user logging, private volumes, and data communications.

The majority of users in a typical HP 3000 installation will simply have capabilities for interactive access and local batch access. MPE simplifies the assignment of user capabilities by establishing a set of default capabilities. If a user needs additional capabilities later, these can easily be added.

The presence of capability sets greatly simplifies the use of the system from the standpoint of each individual user by defining the extent to which he must understand MPE and permitting him to ignore those aspects of the system that do not apply to him.

USER INTERFACE

Three user-oriented software facilities provide a comprehensive interface between the system/application programmer and MPE. These tools are: the MPE command language, the on-line HELP facility, and job control facilities.

Command Language

The simplicity of the MPE command language greatly enhances the system's usability. MPE commands enable you to initiate a session and specify the various MPE operations you wish to have performed. When you specify a command, a portion of MPE called the command interpreter reads the command, checks its validity, and then causes the appropriate action to be taken. After the requested action is successfully completed, the interpreter processes your next command. You may enter commands interactively during a session or through a batch input device. Commands may also be issued programmatically from a running program.

MPE uses a colon (:) to prompt you for a command during an interactive session. When a batch job is submitted, MPE commands within the job are designated by a colon in column one.

The MPE command language is composed of many commands. Each command enables you to request a specific action of MPE. Collectively they provide a powerful system-usage tool. The full range of MPE commands is presented in Appendix B of the HP 3000 Specification Guide. The following is a list of common command uses:

- Initiating and terminating jobs and sessions.
- Running system programs or compilers.
- Running programs.
- Running system utilities.
- Creating, managing, or deleting files
- Displaying file information.
- Displaying job, session, or device status.
- Transmitting messages.
- Assisting in program debugging.
- Establishing communication between a local and a remote computer.

If the command interpreter detects an error during a session, MPE informs you with an error message which specifies the erroneous parameter. MPE then requests that a new command be entered. You can instantly correct command errors by retyping the command or using the REDO command, which allows you to edit the erroneous command, and the session can continue.

During a batch job, MPE lists the error on the listing device. Input from that point through the next EOJ, DATA, or JOB command are usually ignored. You can, however, use the CONTINUE command to request that the job be be continued despite the error.

User-Defined Commands

MPE allows you to define your own commands by combining several MPE commands into a command procedure and assigning the procedure a name. The name can then be used as a command. Thus it is possible to enter a single command name which you have defined and cause several commands to be executed. These user-defined command sets can be created by each individual user as well as being made available to entire accounts and all accounts systemwide. It is also possible to redefine existing MPE commands and messages to suit your particular situation.

On-line HELP Facility

Whenever you need assistance with command language syntax or even the name of a particular command, you can invoke the on-line HELP facility. This facility provides graduated information on any MPE command or set of commands. The HELP messages displayed coincide with the information contained in *HP3000 Command Reference Manual.*

Figure 5-3 demonstrates the two ways in which the HELP facility can be used. In the "immediate" mode, you merely enter the HELP command followed by a parameter. Information detailing that parameter is displayed immediately. In the "subsystem" mode you enter the HELP command without any parameters. The system then displays a menu of valid parameters, and prompts you with a greater than sign (>) for the parameter you wish explained.

```
:HELP REMOTE,EXAMPLE ———————— Example of HELP being used in the "immediate" mode.

EXAMPLE

To establish the communications link HDS2 and log on to System B
from System A, you could enter:

   :HELLO USER.X
   :REMOTE HELLO USER.X;DSLINE=HDS2
KEYWORDS:PARMS,OPERATION,EXAMPLE
: ———————————————————— MPE Prompt. You are given a new prompt automatically
                        following the display of information.
```

```
:HELP ———————————————— Example of HELP being used in "subsystem" mode.

Information is available on the following classes of commands:
  Running Sessions
  Running Jobs
  Managing Files
  Running Subsystems and Programs
  System Management, Status, and Accounting
  Utility Functions

For more information enter a KEYWORD.  You can also enter any
command name as a keyword,  Enter "help" for information on help.
Enter "exit" to leave help.
KEYWORDS: SESSIONS,JOBS,PROGRAMS,FILES,MANAGE,UTILITY
>SESSIONS
Running Sessions.  Following are the commands used:
  ( ) COMMAND LOG ON
  ABORT
  BYE
  DSLINE
  EOD
  EOF
  HELLO
  HELP
  REMOTE HELLO
  RESUME

You can use any command as a keyword.
KEYWORDS:SESSIONS,JOBS,PROGRAMS,FILES,MANAGE,UTILITY
>EXIT ———————————————— You EXIT the HELP Subsystem
: ———————————————— MPE Prompt
```

Figure 5-3. Using the on-line HELP facility.

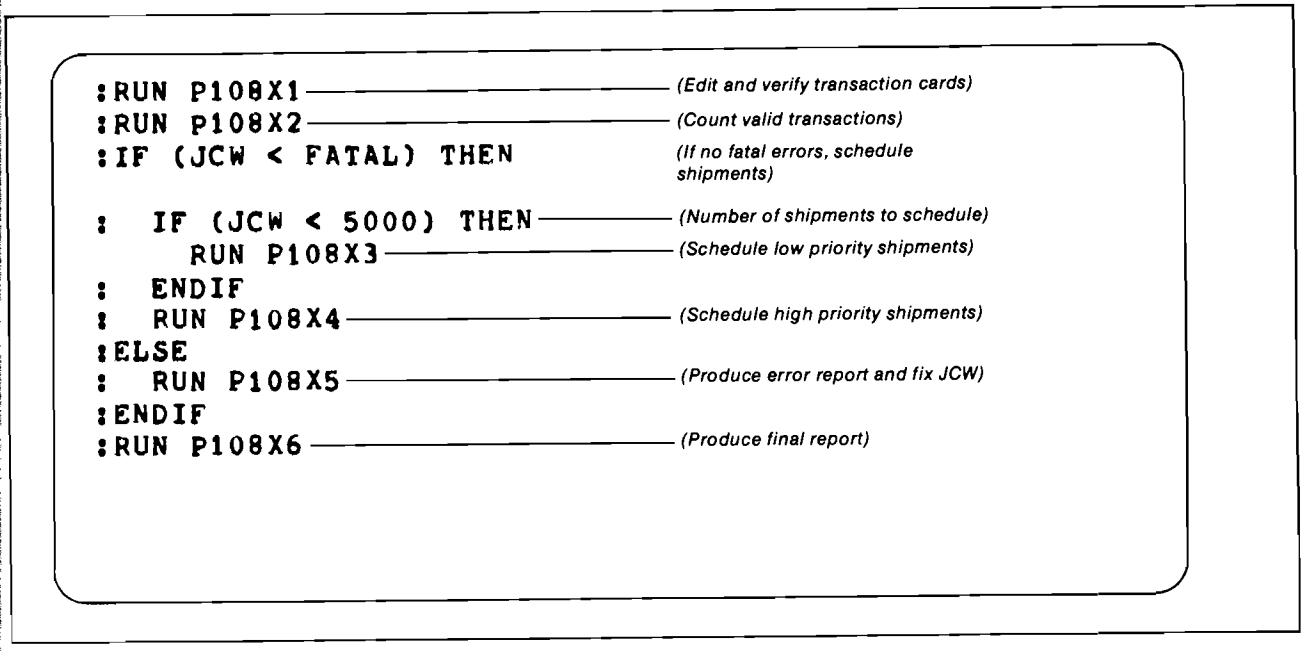
Job Control Facilities

MPE contains job control words (JCW) and conditional execution functions which permit the user to design job streams whose execution can be dynamically altered based on the results of previous job steps.

You can use both system defined and your own job control words to store job status information and to pass such information between programs and between a program and the MPE command interpreter. JCWs are defined and accessed by commands from the command interpreter and by intrinsics from your programs.

You can also use JCWs in conjunction with conditional execution function statements. These statements specify a logical expression (TRUE or FALSE), and are evaluated during program execution. If the value found is TRUE, the remaining statements related to that condition are executed. If the value is FALSE, any existing alternative statements are executed instead.

The following example illustrates the use of JCWs and a conditional execution function. The sample job runs a program which edits and verifies transaction cards and counts valid transactions. If no fatal errors are encountered, the job schedules shipments (either all shiprnents or only high priority shipments depending on the value of JCW) and produces a final report. If fatal errors do occur, the job does no shipment scheduling. Instead, it produces only an error report and a final report.



PROGRAM DEVELOPMENT

The Multiprogramming Executive provides meaningful assistance with the task of generating application programs. MPE programming assistance includes:

- Consistent command language interface to all compilers.
- Program preparation performed by the MPE segmenter.
- Procedure libraries for external references.
- A device-independent file system.
- Flexible file security.
- Subroutines callable across languages.
- Access to all system intrinsics

The use of these software tools during program generation is described below.

Creating Programs

Three steps are required to take a program from source form to an executable state. The first step is to compile the source program into relocatable binary modules called RBMs. This is done by the various MPE language compilers which automatically store the RBMs in a specially formatted file called the user subprogram library or USL.

The second step is to take the USL file and prepare it into a program file. Program preparation resolves external references and results in loadable code segments. This step is done by the MPE segmenter.

The third and final step is to have the MPE loader allocate entries in the HP 3000 code segment table for all the segments in the program file, and to allocate an entry in the HP 3000 data segment table for this process data stack.

Often all of these steps are initiated by single MPE command. (This was illustrated in Figure 5-2.) When necessary, however, you can initiate each step individually, thereby controlling what happens along the way.

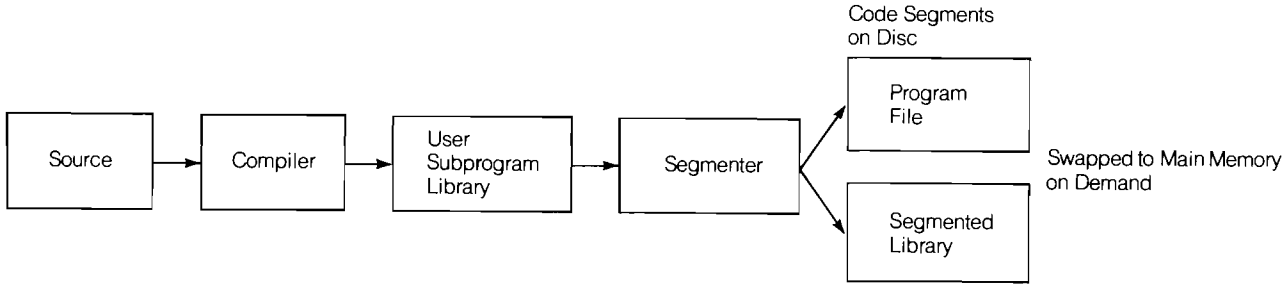


Figure 5-4. Code segment evolution

Accessing Compilers

Program compilation is the first step in converting a source program to an executable state. The format of the commands used to access a language compiler is consistent for all the MPE language compilers. Thus you do not have to learn a new method of program preparation for each programming language you employ.

Three commands are used in the process of program preparation. The first command compiles the program only and stores the resulting RBMs in a USL file for later use. The second command compiles and prepares the program, creating a program file for use in program execution. The third command compiles, prepares and executes the program.

The compiler name is used in the format of the commands. To illustrate, the commands for accessing the COBOL compiler are:

COBOL which compiles the source program

COBOLPREP which compiles and prepares the program

COBOLGO which compiles, prepares, and executes the program

Access to the other compilers is identical, except that the name "SPL," "BASIC" "RPG" and "FORTRAN" replace "COBOL" in each of the three commands. The command BASIC invokes the BASIC interpreter, whereas the command BASICOMP invokes the BASIC compiler. APL invokes the incremental APL compiler

It is important to note that the data files created by these languages are all generated by the same MPE file system. Thus these data files are shareable among the various languages. For instance, a file created by a BASIC program can be read by a FORTRAN program. This file-sharing characteristic also carries down to many HP 3000 subsystems such as KSAM and IMAGE.

Segmenter

Program preparation is actually performed by the MPE segmenter in response to a PREP command or one of the combining forms which include PREP or GO. Segmenter commands may be used to manage USL files by adding, deleting, activating, or deactivating RBMs within them and to build and manage segmented libraries (SLs) which are used to resolve external references from user programs.

Occasionally, you may wish to alter the segmentation of a program to improve its run time efficiency. In many other systems the program would have to be recompiled. With MPE, however, the segmentation can be modified by using the segmenter to rearrange RBMs and then preparing the USL file into a new program file.

Another feature of the segmenter allows different versions of the same subprogram to be stored within a single USL file, and an optional index capability of the segmenter lets you activate and deactivate entry points within various versions of the subprograms.

Procedure Libraries

When a program is allocated and scheduled for execution, MPE searches the following libraries for unresolved external references:

1. The user's log-on group library
2. The public group library of the user's log-on account
3. The public group library of the system account.

Each library can possess two types of library files: segmented library files, and relocatable library files. Segmented library or SL files contain procedures in segmented form which may be shared between programs. Relocatable library or RL files contain procedures in RBM form which must be prepared before they can be loaded with your program.

Procedures contained in the SL file are in prepared form, that is, they are segmented. When a particular procedure is needed, the segment containing the procedure is loaded, as are all external references from that segment. Because the segmentation has been pre-defined in this manner, these procedures may be shared between programs, and only one copy will exist at any time in virtual memory even though several users may require a particular procedure concurrently.

The combination of segmented libraries and relocatable libraries gives great flexibility for storing often used subroutines and procedures. Procedures used system wide are normally stored in the segmented library at the system level. Procedures used by only a few users or a single group are stored in relocatable or segmented libraries at the group, account, or system level.

Special segmenter commands are provided by MPE which enable you to build an SL or RL within a particular group. In addition, commands are available to add or purge routines, and list the procedures contained in either library.

Code Segmentation

To fully appreciate the programming assistance provided by the segmenter, it is necessary to understand the logic behind program code segmentation. In the HP 3000 computer system program code is grouped into topical entities which consist of one or more procedures or subroutines. Each code segment may be up to 32 k bytes in length (where k=1024). Programs may be broken into multiple segments with procedures or subroutines fully contained within one segment.

A code segment consists entirely of information that is not subject to change during program execution. This includes program instructions and constants. No modifiable data may be interspersed with the instructions in a code segment, and it is only possible to change a code segment by recompiling the modified source. This feature ensures that all code is re-entrant, meaning that any sequence of instructions can be in simultaneous execution by multiple users. All HP 3000 computer system procedures are potentially recursive.

The fact that code segments are not modified during execution has specific advantages for the memory management system. Since all code segments are reentrant, all are potentially shared by more than one user when present in main memory. For example, operating system services are provided in part by segments contained in a system segmented library shared by all programs which request those services. Similarly, all users executing the same program file share the program's segments. Only one copy of a segment needs to be in main memory, no matter how many users may be executing it concurrently. Thus the system is able to handle more users in a given memory capacity.

Another advantage of code segment re-entrancy is that code segments are read-only. Thus they never need to be swapped from main memory back to disc, even when overlayed, because there is always an identical copy of the segment in the program file or library. Code is swapped only into main memory, never out. The resulting reduction in swap traffic leads to more efficient memory management.

File System

One of the main uses of a computer system is information management, i.e., the input, processing, and output of data. MPE manages information by means of its file system. With MPE a file is a body of information or data identified by a user-assigned name. A file may contain commands and/or programs, as well as information, and may be stored on disc, tape, or cards.

MPE also treats peripheral devices as files. Access to such files is device-independent, meaning that a program can read data from a card reader, terminal, magnetic tape, or disc by means of the same request. MPE automatically locates, buffers, transfers, and deblocks the data.

When you ask to read a named disc file, you are only implicitly specifying the disc address of the file; the MPE system determines the explicit address and performs the read. In the same manner, when you ask for a certain type of device by specifying a device class name (disc, line printer, etc.), the file system allocates the actual device for you.

The MPE file system permits sequential access to all files. Disc files with fixed- or user-defined length records may also be accessed randomly. Extensive disc file back-up facilities are provided for all types of disc files. The STORE command copies files to a serial storage device; the RESTORE command restores the files to disc.

Files can be accessed from any programming language by means of standard MPE file system intrinsics. A file can be accessed simultaneously by multiple programs; MPE automatically resolves any contention problems which might occur.

MPE file system commands enable programs to reference files without specifying their actual names, addresses, or characteristics. A file can be redefined without a major change to the program. For example, a program's input file named CARDIN designated as a card reader could be changed to a disc file through the use of a FILE command. File specifications can also be altered at run time by means of commands. To illustrate, a program could be coded to open a file with a record length of 128 characters. If at run time it is determined that the file has only 64 characters, you can override the file opening with a FILE command that designates the file to be 64 characters. Reprogramming is not required.

MPE file system commands can be used to build, purge, rename, and display file characteristics. You may specify how disc space is to be dynamically acquired, whether to deal with logical or physical records, whether to include special characters and so forth.

The user logging facility of MPE allows users and subsystems to record additions and modifications to files. In the event of a loss of a file, the user logging record can be used to recover the data in the file. In addition, the logging file can be used as a record of the activity in that file. The entire user logging facility is implemented through the use of MPE commands and intrinsics designed for this purpose.

The MPE file system is actually a collection of routines which reside in the system segmented library (SL). These routines enable you to open a file, obtain status information, read or write data, perform control functions, and close the file. When a program contains statements or constructions that input or output data, these procedures are brought into play automatically by MPE. The loading operations done by MPE to run your program search the library and establish linkages to allow these routines to be referenced during program execution. The code segments containing these file system procedures are shareable, as are all code segments under MPE, and may be used by several programs at the same time.

Subroutine Compatibility

The MPE file system allows programs written in one language to call subroutines written in another language. Once the subroutine is written in the chosen language, it is stored in the system SL (segmented library). Then, user programs written in FORTRAN, BASIC, or SPL can access this subroutine with a standard subroutine call. COBOL and RPG programs can also execute these SL subroutines via an SPL subroutine call.

This ability to intermingle different programming languages significantly expands programmer productivity and application efficiency. Your programmers are free to program in the languages that they are most familiar with. The different language blocks can then be linked by a master program for execution. This way, segments of programs can be written in the language that is most efficient for the operation being performed. This MPE ability allows you to expand the efficiency and capabilities of your application programs.

File Security

MPE provides two general methods of file security. The first is the use of passwords. The creator of a file can establish passwords (also referred to as lockwords) which must be correctly supplied when anyone makes reference to that file. The second method of file security is the use of file access mode and user type restrictions as outlined in Table 5-2.

The system manager specifies the file access modes allowed for an account and the types of users to whom they are available. The account manager specifies the access modes allowed for a group and the types of users

to whom they are available. Finally, the creator of a file specifies the file access modes allowed for the file and the types of users to whom the file is available.

In this manner, access to files can be controlled at several levels which range from unrestricted access making the file available to anyone, to controlled access making the file available to its creator only. For example, you can make your data file available to any other user in a "read-only" mode, while only members of your account can append data to the file.

Often a need exists to save general purpose utility programs in public groups or accounts which may be accessed by all system or account users. MPE provides a special system account named "SYS" and a public group named "PUB" which exists under any account with less-restrictive default security provisions.

Intrinsics

A multitude of additional system functions are available in the form of special MPE procedures, called intrinsics, which may be invoked by calls from your program. Intrinsic calls are acted upon when the segmenter prepares the program containing the intrinsic calls for execution. The segmenter establishes a link between the executing program and the MPE procedure specified by the intrinsic call.

System intrinsics are written in the HP 3000 Systems Programming Language (SPL) and follow the rules and constraints of that language. They may be called from COBOL, BASIC, FORTRAN, or SPL programs, and from RPG programs by way of an SPL subroutine.

There are MPE intrinsics for:

- Opening and closing files
- Reading from, writing to, and managing files
- Controlling devices (such as rewinding magnetic tapes)
- Obtaining file information
- Obtaining user information
- Obtaining detailed error information
- Performing data translation
- Obtaining date and time
- Process handling
- Resource handling
- Data segment handling
- User logging

ACCESS MODES	
Reading	Allows the user to read files.
Appending	Allows the user to add information and disc extents to existing files.
Writing	Allows the user to delete or change information already present in existing files.
Executing	Allows the user to run programs stored in existing files.
Locking	Provides a logical lock which gives the user exclusive access to a file if desired.
Save Files	Allows the user to save permanent disk files within the user's group.

USER TYPES	
Any User	Makes the specified access modes available to any user of the system.
Account Member	Restricts the specified access modes only to users of the account in which the file resides.
Account Librarian	Restricts the specified access modes only to the account librarian of the account in which the file resides.
Group User	Restricts the specified access modes only to users of the group in which the file resides.
Group Librarian	Restricts the specified access modes only to the group librarian of the group in which the file resides.
Creator	Restricts the specified access modes only to the creator of the file.

TABLE 5-2. MPE FILE ACCESS MODES AND USER TYPES

Appendix C gives a complete list of all MPE intrinsics with a brief description of each.

When a system intrinsic is invoked to perform a system function, two types of error conditions may occur. MPE informs the calling program of a recoverable error by setting the condition code bits of the HP 3000 status register when the intrinsic is exited. The condition code indicates whether or not the request was granted and what conditions existed pertinent to the request. A request to an intrinsic which requires a special capability class not possessed by the calling program, or which passes illegal parameters to an intrinsic, is considered an irrecoverable error and causes the system to abort the program. In such a case, if you have not specified an appropriate "trap procedure," a batch job is usually removed from the system; an interactive session resumes with a message and a prompt for another command. The CONTINUE command can be used to continue execution of a batch job despite an irrecoverable intrinsic error Also, by initially calling system trap intrinsics, you may specify special actions to be taken in the event of an irrecoverable error.

A DYNAMIC ENVIRONMENT

The Multiprogramming Executive environment is a dynamic one where programs are run on the basis of processes. A process is the basic executable entity of MPE. It is not a program, but the unique execution of a program by a particular user at a particular time. MPE automatically creates, manages, and deletes processes.

When you execute a program, a private hardware protected data segment called a stack is created for that particular execution. The stack and the program's code segments together constitute the process. To illustrate, when multiple users access the BASIC interpreter, a separate process is created for each of them. They all use the same code, there is only one BASIC interpreter; but each has his own data stack (environment) created by MPE.

The creation, maintenance and deletion of processes is accomplished by means of three MPE components: the virtual memory manager, the job/session scheduler, and the process dispatcher.

Virtual Memory

MPE's virtual memory manager uses both main memory and disc storage to greatly expand the total amount of memory space available. In fact, virtual memory allows programs up to 2 megabytes in length to be executed in minimal memory systems (256Kb). MPE logically divides programs into variable length segments of code and data. These segments reside in disc memory and are brought into main memory only when required for program execution as shown in Figure 5-5. When a code segment is no longer needed, it is overwritten by a new segment. If the code segment is needed again later, it is simply copied again from the disc on which it resides.

Data segments are dynamic and are handled somewhat differently. Since the content of a data segment may change during program execution, a data segment is copied automatically back to the system disc when it is no longer needed, thereby replacing the previous version of that segment.

This approach of segmenting code and data and transfering the segments back and forth between main memory and disc memory results in the allocation of local storage only as needed. In addition, since program code segments are not modified during execution, multiple users are able to share a single copy of the program code.

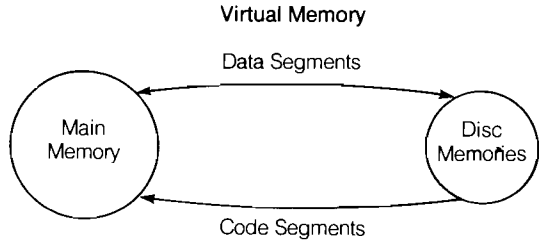


Figure 5-5. MPE virtual memory

A closer look at memory management identifies additional advantages. MPE divides main memory into two areas. The first, fixed memory, contains only those items required to be memory resident such as interrupt handlers, the memory manager, and the scheduler. The remainder of memory, linked memory, contains all other code and data. User and operating system segments are brought into this area by the memory manager as they are required. The architecture allows the operating system, including the file system, the command interpreter, the spooler, and even much of the I/O system, to be shared by all users even though they are brought into main memory only when needed.

The MPE memory manager is responsible for the allocation of main memory to the executing processes. Program and library segments which are needed for execution are automatically swapped into main memory from disc. An attempt by an executing process to access code or data not present in main memory causes the memory manager to allocate main memory space for the missing segment. When the absent segment is swapped from the disc memory to the main memory, the executing process continues. Frequently used segments remain in main memory, and may never be swapped, while rarely used segments are in disc memory most of the time. This results in higher efficiency and faster overall execution time. It also creates a dynamic situation in which segments are being swapped rapidly between main memory and disc memory, according to the demands of the executing programs.

Automatic Scheduling

The MPE job/session scheduler schedules jobs and sessions according to their assigned priorities. When the execution of one process is interrupted for any reason, such as I/O or an internal interrupt, control is passed to the process with the next highest priority which is awaiting CPU resources. When two or more programs have the same priority, the oldest process is selected first.

Jobs and sessions are scheduled by means of a master queue which is ordered by priority as shown in Figure 5-6. This master queue is divided into areas called priority classes. Each area is bounded by two priority numbers established by the system manager.

MPE automatically assigns priority classes to each process executing on the system. The user may, however, specify priority classes by selecting a general category of process dispatching priority for the program. This is done by including the PRI = parameter in your JOB or HELLO command. The five process dispatching priority types (queues) available are:

AS —system processing only

BS —very high priority

CS —interactive

DS —batch

ES —very low priority (background)

MPE actually translates priority types into numerical ranges which are ordered in a master queue (see Figure 5-6). The numerical range of each priority type can be changed at any time to ensure that an optimal balance of services is maintained among the processes on the system.

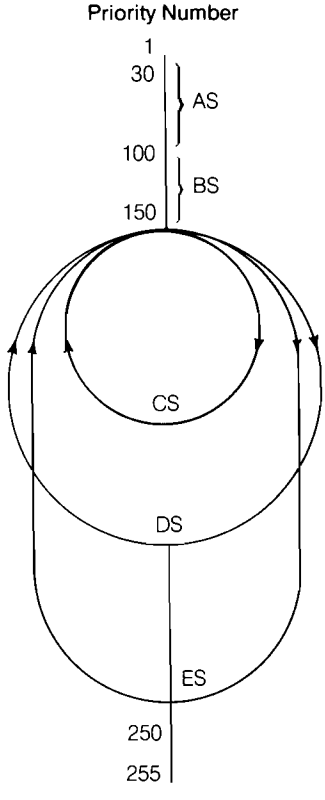


Figure 5-6. MPE master queue structure

Process Execution

The MPE process dispatcher schedules processes for execution. Each process has a dynamically changing priority number, and the dispatcher keeps a list of active processes (those requesting execution) ordered by priority. The dispatcher attempts to execute the highest priority process first. If that process is not in memory, the dispatcher instructs the memory manager to make enough of the process's segments present in memory to allow it to run.

As a process runs it may require another code or data segment. If the segment is not present in main memory, the memory manager is instructed to retrieve the segment before the process is allowed to continue executing. While the process waits for the needed segment to be transferred, MPE transfers control to the next process ready to be executed.

The memory manager determines the working set for each program. The working set is the particular set of segments required in memory for a process to run efficiently. The working set is determined by means of MPE's "segment trap frequency" algorithm, and the size of the working set is expanded or contracted as necessary by MPE. If a working set is too large, the process will control more main memory than it actually needs and thus degrade the performance of other processes. If a working set is too small, the process itself will run inefficiently.

When extra segments are transferred to main memory for a process, the dispatcher determines whether or not to add them to the working set of that process. When segments are removed from the working set, they simply reside on disc.

If there is more main memory available on the system than is required for the working sets of all active processes, code and data segments are left in main memory. If there is insufficient space in main memory for a new segment request from an active process, the memory manager creates space as follows:

- First, available free areas are used.
- Next, segments available for overlay are located and used. Overlaying a data segment involves ensuring that the segment has been copied back to virtual memory.
- The space created by the removal of overlayed segments is coallesced with any other free space available.
- If additional space is needed after all overlayable segments have been used, part of the working set of an active process is temporarily removed by the dispatcher and its space is used for the requested segment. The process with the lowest priority is always selected.

The objective of the process dispatcher and the memory manager is to provide for optimum efficiency in the use of system resources while satisfying the requirements of executing processes. This is done automatically by MPE without assistance from the system users.

Privileged Mode

Privileged mode operation lets you access all MPE resources directly, including privileged intrinsics, system tables, and privileged CPU instructions. In essence, the privileged mode option lets you function as the operating system.

Normally your programs are not able to affect the operation of other users or of MPE itself. Privileged mode operation, however, bypasses normal operating checks and limitations. It is possible for a privileged mode program to destroy file integrity, including that of the MPE operating system itself. Hewlett-Packard can not assume responsibility for system integrity when your programs operate in privileged mode.

SYSTEM OPERATION

This section deals with the day-to-day operational aspects of running an HP 3000 installation. For all of its power and features, MPE is surprisingly easy to maintain. General operation of the system is primarily the responsibility of three users: the system manager, system supervisor, and the console operator. The console operator is responsible for routine day-to-day system operations. It is his task to respond to system messages and to keep the system and the peripheral devices functioning smoothly and efficiently. The system supervisor, who is appointed by the system manager, has day-to-day responsibilities in several areas. Included in his duties are the maintenance of the system logging and resource accounting facilities. He also has the capability to retrieve information and change parameters relating to the master scheduling queue. The system manager implements MPE's unique account/group/user organization and appoints account managers to monitor account usage.

System configuration, the first operation undertaken with a new HP 3000, establishes the boundaries under which MPE will function in terms of memory and peripheral devices. An interactive process that takes place with the operator at the system console, system configuration requires intervention only when options other than the system defaults provided are specified, and can be accomplished in a matter of minutes. Once configured, the system is ready for use.

The next operation involves the building of various accounts and the identification of system users. Usually, the first user identified is the system manager who will have overall control of the system. Among his responsibilities is the task of allocating various accounts within the system and identifying an account manager for each. The account managers in turn identify the users who may access the system through their respective accounts.

Once the system is configured and the account and user structure has been defined, the operation of program development begins. Programming may be done interactively on the HP 3000 by means of terminals and the many programming aids provided by MPE for the development of source programs; or programs can be developed in batch processing mode. Once developed, programs are executed as frequently as required.

Occasionally the account manager may need to add new users to their accounts, or it may be necessary to reconfigure the system as new peripherals or more memory is added. But the primary continuing operations are the development of programs and their execution. Figure 5-7 illustrates the interrelation of these general HP 3000 system operations.

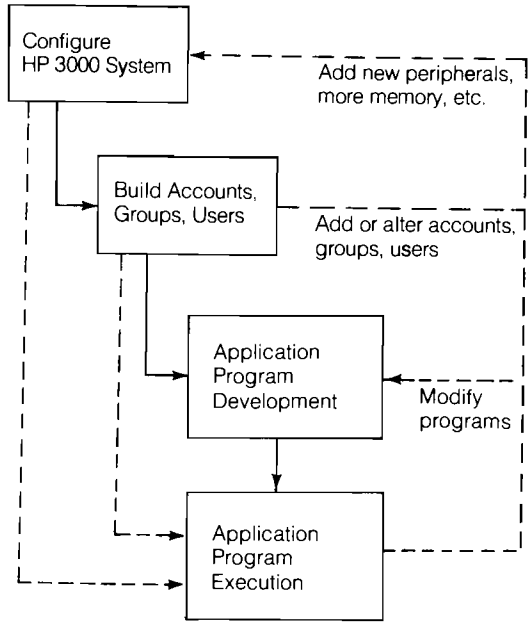


Figure 5-7. HP 3000 operations

Console Operator Function

The console operator function is available to any one user logged onto a non-REMOTE interactive terminal on the system. Operator commands are entered in SESSION mode just as are MPE commands. The console operator has the ability to move the console to other terminal devices, and may selectively allow users to execute specific operator commands. The operator can also allow individual users to have operator control over specific devices. Operator commands deal primarily with peripheral and other hardware system requirements, as well as with system start-up, back-up and recovery procedures.

Start-up and Modification

MPE is initially brought up on an HP 3000 by the operator at the system console, through one of several restart operations:

- WARMSTART—The system is restarted from the disc. Spoolfiles are recovered and incompletely processed spooled jobs are automatically restarted.
- COOLSTART—The system is restarted from the disc. The spoolfiles in existence when the system was shut down are not saved, but all resident user files are saved.
- COLDSTART—The system is read from a serial storage device (magnetic tape or serial disc). The I/O configuration and system configuration from the serial storage device used to define the system are merged with the directory and files present on the disc.
- UPDATE—Similar to cold start, except that the I/O configuration on the disc is used to define the system. This start-up mode is used when starting the system from an updated version of MPE supplied by Hewlett-Packard.
- RELOAD—The entire system, directory, files, and configuration information are read from a serial storage device.

The restart operation can include an interactive dialogue between the console operator and the MPE initiator program. This optional dialogue permits the operator to change the system configuration. Upon completion, MPE is operational.

System Backup and Recovery

Periodically, the MPE system and user files are copied on a serial storage device for back-up purposes. The serial storage device is then available for reloading the system in the event of a hardware or software failure, or to transfer the system to another hardware installation.

In the latter case, a new I/O and system table configuration may be specified during an interactive dialogue. Areas which may be changed include:

- I/O devices
- System table sizes
- System disc allocation
- Logging facility
- Scheduling changes
- Segment size limits
- System modules to be allocated or made memory resident

The SYSDUMP command (which requires System Supervisor capability) may be used to create media for reloading on a different or changed HP 3000. A new I/O configuration and system table configuration may be specified during the interactive dialogue which occurs. A back-up date may also be specified which enables the operator to back-up only those disc files which change each day.

Power Fail/Auto Restart

When an AC power failure is detected by the HP 3000, a power fail/auto restart routine is automatically invoked. This routine preserves the operating environment prior to complete loss of power. Normal system operation resumes as soon as power is restored. Jobs and sessions in progress on the system continue where they were interrupted, with programs unaware of the interruption (except for magnetic tape units and dial-up terminals in use when the failure occurred, or if the power outage lasts longer than the built-in system memory battery backup).

In addition to the system configuration aids mentioned above, MPE provides a range of software aids specifically designed for use by the system operator. Among these are:

- Spooling facility
- Tape label facility
- Disc options.

Spooling Facility

This MPE facility permits the concurrent usage of devices which would otherwise be non-shareable, such as card readers, magnetic tape drives, or line printers.

This is accomplished by copying the input or output to disc where it is processed later. This procedure is called spooling. (SPOOL is an acronym for Simultaneous Peripheral Operations On-line.)

To illustrate, if six users need to produce output on a line printer at approximately the same time, their output is directed to spoolfiles on disc from which the output is printed on a priority basis as the line printer becomes free. In this way each user can immediately proceed with other processing activities without having to wait for the line printer. Similarly, if there are ten jobs to be read from a card reader or magnetic tape unit, they are all read immediately and are directed to spoolfiles on disc where they wait to be executed. Thus, the card reader or magnetic tape unit is not tied up by one job which must be executed before the others can be read.

The spooling of batch job input can be initiated not only from a card reader or magnetic tape unit, but also from within an interactive session as described earlier.

Tape Labeling Facility

MPE provides a tape labeling facility for use in reading and writing labels on magnetic tapes. The facility can be used to:

- Identify magnetic tape volumes (reels).
- Protect tape volumes from being inadvertently written over.
- Protect private information.
- Facilitate information interchange between computer systems.

The facility can be used to read, but not write, IBM-standard tape labels; read and write ANSI-standard labels; and read and write user-defined labels on previously labeled magnetic tapes.

Disc Options

The MPE operating system includes a serial disc interface which allows non-system domain drives to be used as non-shareable serial devices. To MPE, the discs appear to be magnetic tape drives. They provide a fast system backup and recovery capability when used as an alternative to magnetic tape in backing up the MPE system and storing and restoring files and/or data bases.

MPE also provides a private disc volume facility which allows you to create and access files on removable disc volumes. Private volumes consist of removable disc packs which, when mounted on a disc drive, can be accessed by MPE through the file system. Under private volumes, the disc packs mounted on the drives during a cold load are dynamically allocated to the system domain for normal use or to the non-system domain for private use. Non-system domain packs can be both physically and logically mounted and dismounted during normal system operation.

System Account Structure

The primary responsibility of the system manager is to create and maintain the organizational structure of accounts, groups, and users under which access to MPE occurs. The account structure provides maximum security and control by permitting the system manager to assign specific access and system usage capabilities to each user.

"Accounts" are collections of users and groups. Each account has a unique name and an optional password assigned to it when the system manager creates the account. Each account also has its own file domain or unique set of files. The system manager may define resource-use limits for an account. MPE maintains a running count of each resource that the account uses. MPE also stores a list of user name and group names recognized by the account, the maximum job priority at which jobs in the account may be scheduled, and limits established on the account's usage of disc file space, CPU time, and connect time.

"Groups" are used to partition the file domain of an account. Files must be assigned to a group, and each group has a unique name (within the account) and optional password. Limits may be established on the permanent disc space, CPU time, and connect time used by a group. MPE maintains running counts of resource usage for each group and the sum of these group counts always equals that of the account in total.

"Users" are individuals who access the HP 3000. Each user is assigned a unique name and optional password, and is assigned to a specific account. Each user may have a specified home group of files, and may access any other file groups in the account. A maximum job priority may be assigned to each user.

Each account "owns" a unique set of files separate and distinct from every other account. This ownership is indirect in that only groups may own files directly. Thus, every file belongs to a group, every group belongs to an account, and every account belongs to the system.

To illustrate how accounts, groups, and users interrelate, consider the following example. Figure 5-8 represents a system which includes interactive terminals dispersed throughout a company. The system manager has assigned three accounts: Marketing, Engineering, and Finance. The marketing account manager has defined two users who can access the system: Bill and Dave. Each user has his private group (assigned as his home group) where he stores his private programs and files. Bill and Dave can also access programs and files stored in the other groups in the account. A group named PROJ1 was created to contain programs and data files related to current projects.

An administrative group was also created to contain administrative work such as schedules and budgets. The public group, to which no password was assigned, contains general purpose utility programs for use by all.

Bill can log on to the HP 3000 from a terminal with the command:

```
:HELLO BILL.MKTG
       user   account
       name   name
```

By default, Bill now has access to all programs and data files in his home group: BILLSGRP. Bill can gain access to a file in the PROJ1 group by using the fully qualified file name which specifies both the account and group under which the file was created. The file Bill wants is data file 1, so he enters:

```
FORECAST.PROJ1.MKTG
file    group   account
```

Alternatively, Bill could have logged on to the HP 3000 and requested access to all programs and files in the PROJ1 group by appending the group name to his log on request, as follows:

```
:HELLO BILL.MKTG,PROJ1
user    account   group
```

Bill now has access to all files in the PROJ1 group.

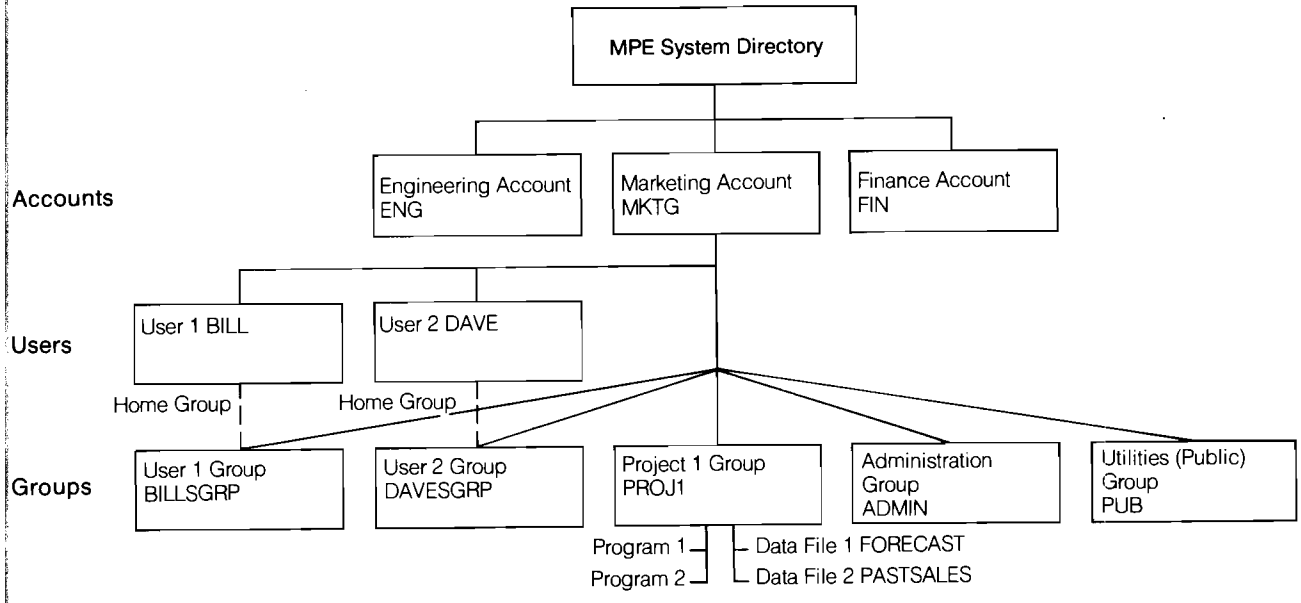


Figure 5-8. MPE account structure

To summarize, you can log on to the system using only your assigned user and account names, in which case you are automatically given access to your home group. Or, you can log on specifying your user name, account name, and a group name which gives you access to the group you specify whether or not it is your home group.

To access a file in a group other than the one specified when you logged on, you use the fully qualified name of the file which can consist of the file name, the group name, the account name, and the appropriate passwords when required.

As you can see, the account structure provides both control and security over file use. Access to the system is granted only to individuals with a valid log-on identification consisting of account, group, and user names, each of which may require a password. Figure 5-9 illustrates both an unsuccessful and successful log-on procedure where passwords are required.

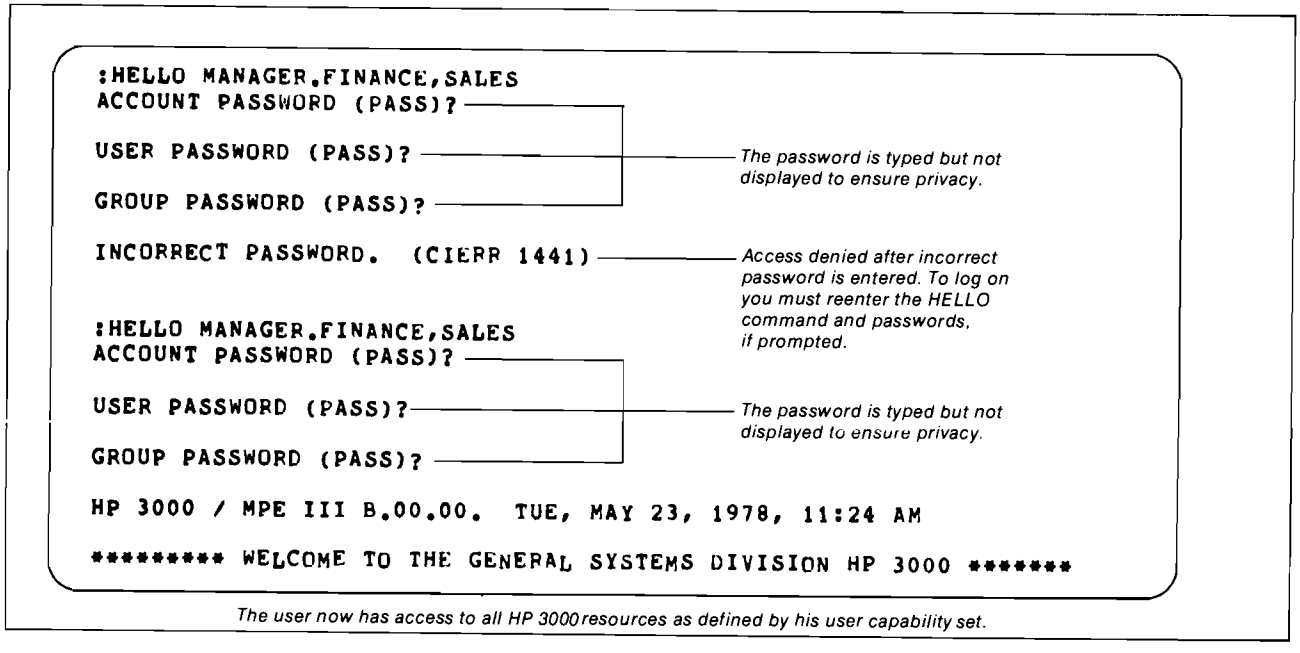


Figure 5-9. MPE system access security

System Supervisor

The user with system supervisor capabilities can use commands which enable him to control the master scheduling queue, permanently allocate or deallocate a procedure or program to virtual memory, and manage the logging facility.

He can control the master scheduling queue in two ways: first by entering the SHOWQ command which displays information about the scheduling of processes and the contents of the master queue and various subqueues; and second, by entering the QUANTUM command which allows changes to the time quantum or limits defining the bounds of any subqueue.

Two important commands, ALLOCATE and DEALLOCATE, allow him to tune the system by permanently allocating a procedure or program in the HP 3000 to virtual memory. This significantly decreases the time needed for an operation with large, frequently used routines.

System Logging Facility

The MPE logging facility records details of system resource requests in a series of log files on disc and can be used to monitor system resource usage. The system supervisor selects those system and user events that are to be recorded. Log records are provided for job and session initiation or termination, program termination, file closing, file spooling completion, and system shutdown. Log files can be used in the generation of precise billings based on accurate system usage records.

I/O device failures on any device are also recorded in log files which can then be used to detect device problems before they begin to interfere significantly with overall system operation.

Accounting Facility

The MPE accounting facility provides a flexible and powerful means of coordinating access to the system and disc file usage. To coordinate system access, system administrators can devise a structure of accounts and users which reflects the functional organization of the people who use the system. The accounting facility maintains running totals on the amounts of system resources that each account consumes, including disc space used, cumulative CPU time consumed, and cumulative terminal connect time for sessions. The current totals can be displayed at any time and can be used for billing purposes.

File usage can be coordinated also because the overall permanent disc file domain of the HP 3000 is partitioned among the various accounts. Each account's file domain is further partitioned into groups. If a request to save a file would result in exceeding the permanent file space limit at either the account or group level, the request is denied.

Users may create files only in the account and group under which they are currently running. By using fully-qualified file names users also have access to any file present in the system, provided that existing security provisions allow them access.

Chapter 6

SYSTEM ARCHITECTURE

STACK ARCHITECTURE
SEPARATION OF CODE AND DATA
PROCESSES
VARIABLE-LENGTH SEGMENTATION
CODE SEGMENTATION
DATA STACK
VIRTUAL MEMORY
MICROPROCESSOR
REGISTERS
INSTRUCTIONS
MICROCODE

Stack Architecture

The HP 3000 with its hardware stack implementation is referred to as a stack machine. A stack is a linear storage area for data. It is so named because data items are placed on the "top," "pushing down" the data items already present. Data items are removed from the top, "popping up" those data items remaining. If you have ever worked with a Hewlett-Packard calculator that has an "ENTER" key, then you have worked with a stack. Consider the following calculation:

$$\frac{6 * 10}{2 * (3+12)}$$

Using a stack, no temporary intermediate values need to be named or stored in registers until required later in the computation. An example of how this problem would be evaluated in a stack is presented in Figure 6-1. Note that the top of stack (TOS) moves downward in keeping with the HP 3000 conventional representation. All operations can be performed without naming specific operands, as the top two stack data values are implicitly used.

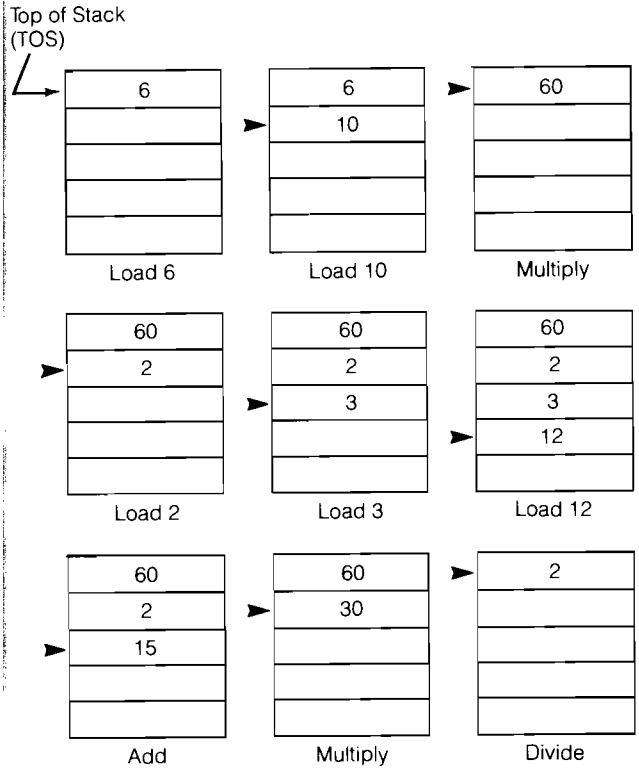


Figure 6-1. Stack Evaluation of (6*10)÷[2*(3+12)]

The benefits of a stack architecture are numerous. First, as seen above, the storage allocation is dynamic. Local storage is allocated only upon entry of a procedure and is automatically freed upon exit so that areas of memory are not tied-up and can be re-used by other parts of the program. Temporary storage of intermediate values is automatically provided. Thus compilers don't have to be concerned with saving and restoring registers for intermediate results.

Code compression is made possible by the omission of operands in many of the instructions on a stack machine. No extra registers are required for subroutine parameters and temporary variables. A register machine requires 50 to 100 percent more bits for code than a stack machine.

The subroutine is one of the most important concepts in software. Modular, structured programming has as its principle idea the partitioning of a large program into many small, understandable modules which can be called as subroutines. The best mechanism for subroutine calls and execution is the stack, because all subroutine return addresses, I/O parameters, and local variables can be pushed onto the stack. This leads to easy parameter passing and provides highly efficient subroutine linkage.

Fast execution on the HP 3000 is a benefit of having several CPU registers to hold the top part of the stack, rather than leaving it all in main memory. The HP 3000 provides each user with hardware protection of his data stack. Rapid interruption and restoration of user environments is made possible by storing the operating environment in a special block as an extension to the user's stack.

Separation of code and data

In most computer systems, programs consist of an inter-mixing of instructions and data. For example, within a subroutine there are program locations reserved by the compiler for return addresses of other subroutines and space set aside for the storage of local variables.

The HP 3000 approach separates a program into those elements that do not need to be altered and those that do. The result is that an HP 3000 program consists of a separate code area and data area (data stack), as illustrated in Figure 6-2.

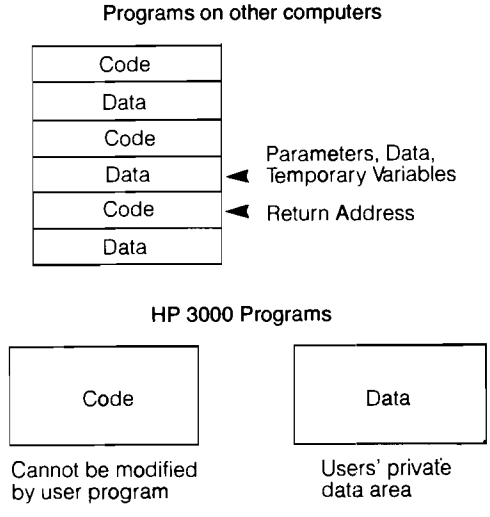


Figure 6-2. Separation of Code and Data

Code consists of the executable instructions that make up a program or subprogram. The values and arrays used by the program or subprogram are referred to as data. Code cannot be altered by your program as there is no instruction available in the HP 3000 instruction set to allow such a modification. However, since you, and you alone, must be able to manipulate your data, the system provides you with a unique, private, modifiable data area (data stack).

In the HP 3000, code and data are maintained in strictly separate domains and cannot be intermixed (with the exception, however, that program constants may be present in code segments). This fact, plus the fact that code is nonmodifiable while active in the system, permits code to be shareable among several users. HP 3000 code is also reentrant. Re-entrant means that when a program is interrupted during execution of a code segment and another user's execution needs the same segment, that segment can be used, is completely protected against modification, and will be returned intact to the previous user's execution. Since re-entrancy allows only one copy of heavily used programs to be shared by many users concurrently, main memory can be used with optimum efficiency. Re-entrancy and stack-structured data together make possible subprogram recursion, a subprogram calling itself, which is essential for efficient compilers and system software. Also, since code is non-modifiable, exact copies of all active code can be retained on disc, thus allowing code to be overlayed without having to first write it back out to the disc.

Processes

Programs are run on the basis of processes created and handled by the operating system. The process is not a program itself, but the unique execution of a program by a particular user at a particular time. If the same program is run by several users, or more than once by the same user, it forms part of several distinct processes.

The process is the basic executable entity. It consists of a process control block (PCB) 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 data in a file and instructions not yet executable, a process is an executing program with the data stack assigned. The code segments used by a process can be shared with other processes, but its data stack is private.

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.

Processes are invisible to the programmer. In normal operation you have no control over processes or their structure. However, optional capabilities are available to permit you to create and manipulate processes directly.

Variable-length segmentation

Variable-length segmentation of code and data is used to facilitate multiprogramming. This method, in comparison with paging schemes, minimizes "checkerboard" waste of memory resources due to internal fragmentation. It also makes it possible for the operating system to deal with logical instead of physical entities. This means, for example, that a particular subprogram can always be contained within one segment rather than arbitrarily divided between two physical pages, thus minimizing the amount of swapping that needs to be done while executing that subprogram. The location and size of all executing code segments is maintained by MPE in a code segment table while the location and size of all associated data segments is maintained by MPE in a data segment table. These tables are known to both hardware and software. Software uses them for dynamic memory management by the operating system. Hardware uses them to perform references and transfers between segments and to make sure that all segments required for current execution are present in main memory. Code segments may be up to 32,760 bytes in length. Data segments may be up to 65,528 bytes in length.

Segments are stored on disc and are brought into main memory only when needed. This design results in a virtual memory environment which appears to be many times larger than the maximum size of the physical main memory. It should be noted that virtual memory in MPE does not contain code segments—only the data segments which must be swapped. The code segments stay in their original positions, anywhere on the discs.

Code segmentation

Code segmentation allows you to divide your program into several segments using control statements to the compiler at compilation time or commands to the MPE operating system. Then the operating system takes over the management of these segments. That is, MPE determines whether or not a code segment should be in main memory or in disc memory. Thus you can write programs much larger than the available main memory of your HP 3000. In the example in Figure 6-3, a program has been divided into five code segments. Code segments 1, 4 and 5 are in main memory while code segments 2 and 3 are in disc memory. The code segment table (CST), which is resident in main memory, points to the active code segments. If the code segment is in main memory, the code segment table points to the beginning of that code segment. If the code segment is on disc, the code segment

table points to its disc address. The management of code segments and the code segment table is completely transparent to your program. When a subroutine call is made from one code segment to the next, the instruction that makes the call examines the code segment table to determine whether or not the new code segment is in main memory. If it is, control is transferred immediately to that code segment. If the code segment table indicates that the required segment is on disc, then the instruction interrupts the operating system and informs it that a required code segment is not in memory. It is then up to the operating system to make space for that code segment in memory for execution, and transfer it to main memory so execution of the process can continue.

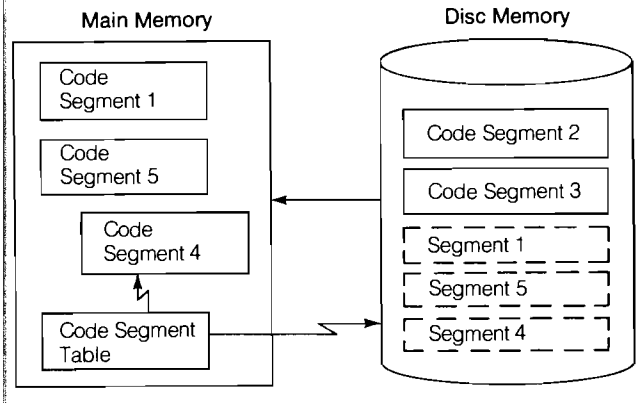


Figure 6-3. Code Segmentation

You have control over segmentation; that is, you determine where the program is divided. The result is that segmentation can be tailored to your program's logic. The size of segments can be optimized to memory size and you can avoid thrashing (unnecessary swapping of code segments due to poor segmentation—across a DO loop, for example).

Data stack

The data area of an HP 3000 program consists of a data stack, an area of main memory that expands and contracts as the program requires. In all programs there is a certain amount of global or common data that is required throughout the life of the program. This global area represents the absolute minimum size of the data stack. Beyond this, the data area grows to meet the needs of subroutines as they allocate storage for their own working areas. When a subroutine has finished its job, this area can be cut back to make use of the same memory space for the next subroutine. The end result is that less memory is required for program execution.

In general, a stack is a storage area in which the last item entered is usually the first item removed. In actual use, however, programs have direct access to all elements in the stack by specifying addresses relative to several CPU registers (the DB, S, and Q registers). The stack structure provides an efficient mechanism for parameter passing, dynamic allocation of temporary storage, efficient evaluation of arithmetic expressions, and recursive subprogram calls. In addition, it enables rapid context switching to establish a new environment on subprogram calls and interrupts. In the HP 3000, all features of the stack (including the automatic transferring of data to and from the CPU registers and checking for stack overflow and stack underflow) are implemented in the hardware.

When programming in a high-level language such as COBOL or RPG, all manipulation of the stack is automatically done for you by the language processor. You can, however, manipulate the stack directly by writing subprograms in SPL (Systems Programming Language for the HP 3000).

Figure 6-4 illustrates the general structure of a data stack as viewed from a subprogram. The white area represents filled locations, all containing valid data, while the shaded area represents available unfilled locations.

You can see that the contents of the data stack fall into four general areas. They are the global data required by all subroutines during the execution of a program, parameters that are passed to subroutines, the local data area required by the currently active subroutines, and the temporary storage required for the evaluation of expressions and intermediate results. The remaining area of the stack is unused and represents the amount of expansion possible in the stack without operating system intervention.

The stack area is delimited by the locations defined as DB (data base) and S (stack pointer). The addresses DB and S are retained in dedicated CPU registers. The Q-minus relative addressing area contains the parameters passed by the calling program. The area between Q and S contains the subprogram's local and temporary variables and intermediate results.

The data in the DB location is the oldest element on the stack. The data in the S location is the most current element. The location S is also referred to as the top of stack or TOS. Conventionally, the top is shown in diagrams downward from DB; this corresponds to the normal progression of writing software programs where the most recently written statement is farther down the page than previous (older) ones.

The area from S+1 to Z (the shaded area) is available for adding more elements to the stack. When a data word is added to the stack, it is stored in the next available location and the S pointer is automatically incremented by one to reflect the new TOS. This process is said to push a word onto the stack. To delete a word from the stack, the S pointer is simply decremented by one, thus putting the word in the undefined area.

To refer to recently stacked elements of data, S-minus relative addressing is used. Under this convention, S–1 is the second element on the stack, S–2 is the third, and so on. S-minus relative addressing is one of the standard addressing conventions. The others are DB-plus relative addressing and Q-minus and Q-plus relative addressing (the Q-register separates the data of a calling program or subprogram from the data of a called subprogram).

Since the top elements of the stack are the most frequently used, there are several CPU registers which may at various times contain the topmost stack elements. The use of CPU registers in this way increases the execution speed of stack operations by reducing the number of memory references needed when manipulating data at or near the top of the stack. These registers are implicitly accessed by many of the machine instructions and whenever the top stack locations are specifically referenced. Data stacks are automatically expanded by the operating system during execution up to a maximum size of 64k bytes (k = 1024).

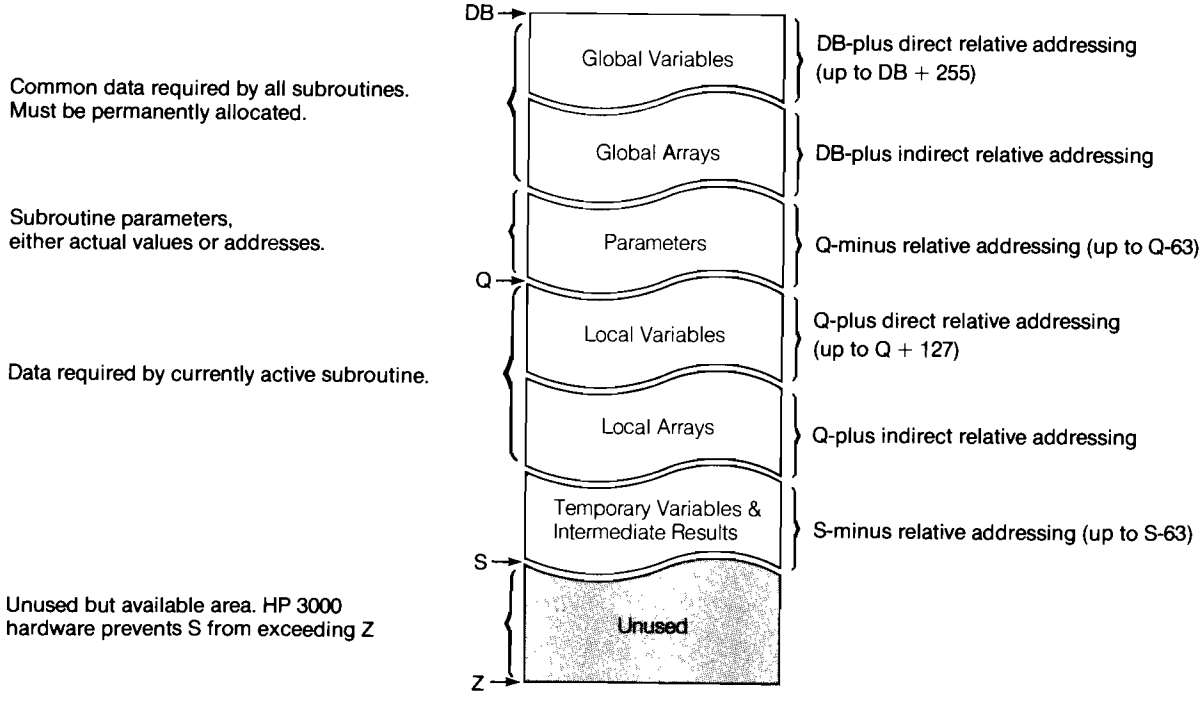


Figure 6-4. Data Stack Contents

Registers

The architecture of the HP 3000 employs a set of specific purpose registers rather than a set of general purpose registers. Each register is included in the system to efficiently perform a single specific function.

All addressing of code and data is done relative to hardware address registers. Thus by simply changing the base addresses in these registers, segments are dynamically relocatable in memory. The few instances where absolute addresses are required are privileged operations handled by the operating system.

Approximately one-half of the HP 3000 registers are accessible to user programs and/or the operating system and its related software. The remaining registers are used, for example, by the interrupt system and for microprogram processing. The registers for the Series 30 and 33 are summarized in Appendix D, and registers for the Series III are summarized in Appendix F in the HP 3000 Specification Guide.

Virtual memory

Virtual memory is a very efficient memory management scheme which, in addition to main (semiconductor) memory, uses disc storage as secondary memory. Users' program code and data are divided into variable-length segments which reside in secondary memory. As a program is being executed, only those segments of code and data which are required at a particular time actually reside in main memory; the other related segments remain on disc until they in turn are required. When a particular code segment is no longer needed, it is simply overlayed by another segment.

This can be done because in the HP 3000, code segments are nonmodifiable and re-entrant. When the segment is needed again, it is simply copied from the disc on which the program resides. Since programs are copied into main memory directly from disc memory, they need not be copied prior to execution to a special "swapping disc." Data segments, however, are dynamic, and their contents can change during execution. Therefore, when a particular segment is no longer needed, it is automatically copied back to the system disc (replacing the previous version of that segment on the disc), and the main memory space of that segment is then available for other segments. The process of transferring segments between secondary memory and main memory is referred to as swapping. Whenever a segment is referenced, the hardware checks to see whether it is in main memory; if it is not, the operating system is invoked to bring it in. Thus the management of the virtual memory is totally automatic and transparent, and the system can reference a virtual memory space far larger than the real memory available.

Microprocessor

The heart of the HP 3000 hardware is the microprocessor. Each HP 3000 machine instruction is mapped into a unique microprogram that resides in control memory and consists of a series of 32-bit instructions that are executed by the microprocessor. One of the first operations in each microprogram is the execution of a NEXT instruction which begins fetching the next instruction from the HP 3000 program. Thus, when the first microinstruction is finished, the next instruction is ready for execution. This parallel execution of one instruction while the next one is being fetched, or pipelining, speeds up the overall execution time.

The primary benefit of a microprocessor is that each instruction does not require its own unique hardware logic. Instead, the instructions share the logic of a common processor called the microprocessor. This means that there is much less hardware involved in order to achieve a very sophisticated and powerful instruction set. Another advantage is that the instruction set is tailored to the compilers and the operating system. Also, the operating system's work is reduced by putting some of the more complicated operations in micro code. For instance, the operation of determining whether or not a code segment is resident in core or on disc is taken care of for the operating system by an instruction. Finally, with a microprocessor, rather than an instruction set implemented in hardware, it is easy to add new instructions. This allows Hewlett-Packard to develop new CPU hardware technology and incorporate it under MPE without changing user software.

Microcode

The entire instruction set of the HP 3000 is in the form of microcoded operations in read-only memory. This microcode is executed by a microprocessor in response to machine instructions fetched into the CPU. By allowing microprogrammed hardware to execute certain repetitive functions, such as subprogram linkage, string processing, and buffer transfers (traditionally software-implemented), the amount of code and execution times are greatly reduced.

In addition to the instruction set, many system operations that in the past were programmed in software have been microcoded. These operations are requested by machine instructions that each, in turn, execute multiple microinstructions built into the central processor hardware. Some of the standard system functions which have been microprogrammed include the interrupt handler, a cold-start loader, the saving of critical environment information on power failure, automatic restart upon restoration of power, and a set of microdiagnostics that can be invoked from the front panel of the system.

The microprogrammed instructions routinely check for addressing bounds violations during execution and automatically interrupt to error handling routines if violations occur. These memory protection checks are usually overlapped with the operand fetch and therefore do not slow execution.

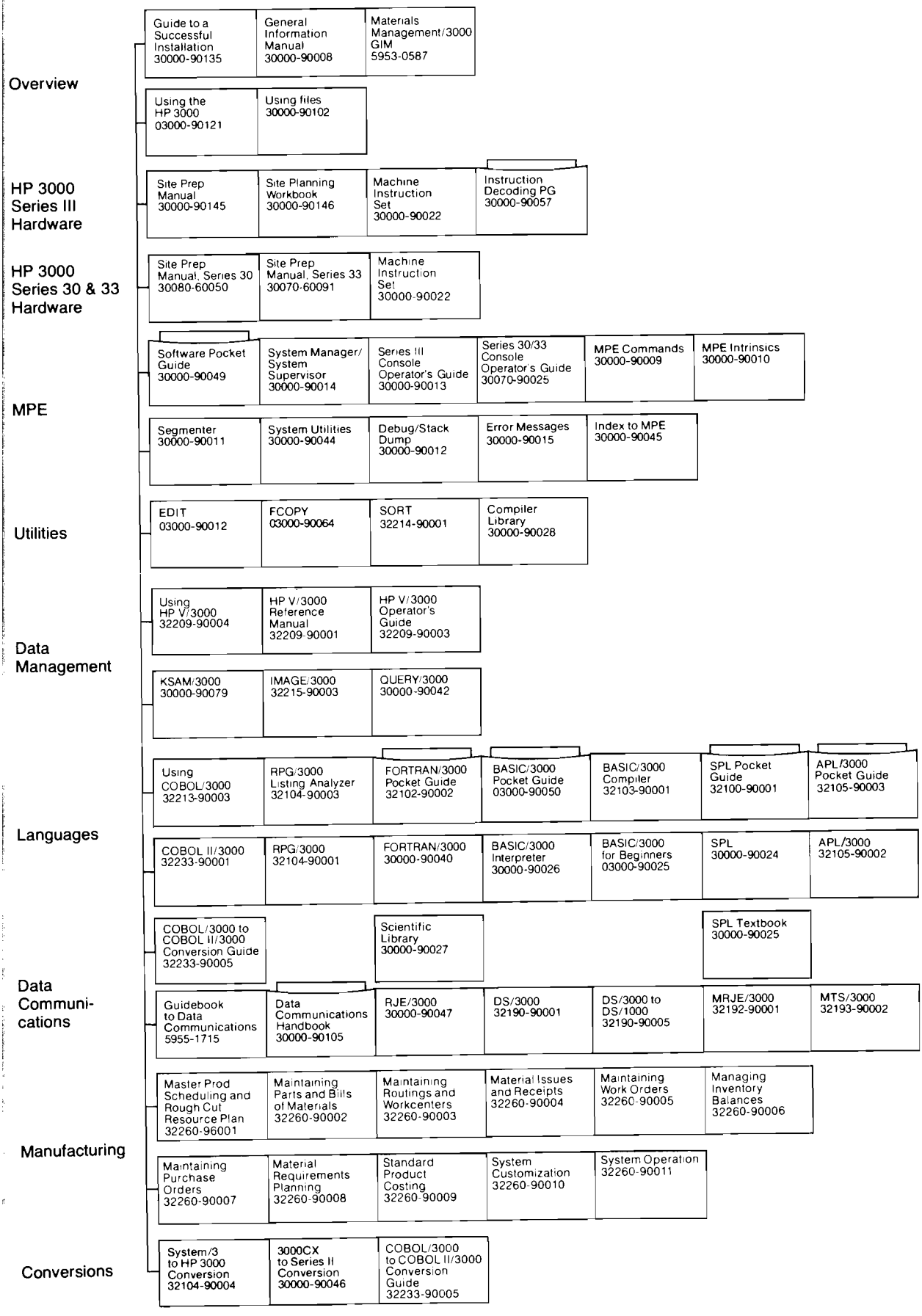
Instructions

There are over 200 unique HP 3000 instructions which are microcoded in a read-only memory (ROM) under separate microprocessor control. Many of these instructions have multiple actions, several addressing modes, indirect addressing, and/or indexing which give a high complexity-to-instruction ratio. Code compression is achieved through the use of no-address (stack) instructions which implicitly use the contents of the stack registers as operands. All instructions except the stack operations are in 16-bit format; the stack operations may be packed two per 16-bit word to further enhance the code density.

A complete set of arithmetic instructions provides integer (16-bit two's complement), double integer (32-bit two's complement), logical (16-bit positive integer), 28-digit packed decimal (BCD coded digits packed two per byte), floating-point (32 bits including a 23-bit precision mantissa), and extended precision floating-point (64 bits including a 55-bit precision mantissa) arithmetic.

Other instructions are designed to facilitate string processing, subprogram linkage, and loop control. Certain special instructions are designated as privileged, meaning that they were designed specifically for use by the operating system. They may, however, be used by programs which the installation permits to run in privileged mode. Some of these special instructions, such as the DISP instruction for entry to the MPE dispatcher, instructions for enabling/disabling process switching, and instructions for data transfers between data segments, contribute greatly to the efficiency of the operating system. Complete machine instructions are provided in Appendix E for the Series 30 and 33 and Appendix G for the Series III.

APPENDICES


Overview
Guide to a Successful Installation 30000-90135
General Information Manual 30000-90008
Materials Management/3000 GIM 5953-0587
Using the HP 3000 03000-90121
Using files 30000-90102
HP 3000 Series III Hardware
Site Prep Manual 30000-90145
Site Planning Workbook 30000-90146
Machine Instruction Set 30000-90022
Instruction Decoding PG 30000-90057
HP 3000 Series 30 & 33 Hardware
Site Prep Manual, Series 30 30080-60050
Site Prep Manual, Series 33 30070-60091
Machine Instruction Set 30000-90022
MPE
Software Pocket Guide 30000-90049
System Manager/ System Supervisor 30000-90014
Series III Console Operator's Guide 30000-90013
Series 30/33 Console Operator's Guide 30070-90025
MPE Commands 30000-90009
MPE Intrinsics 30000-90010
Segmenter 30000-90011
System Utilities 30000-90044
Debug/Stack Dump 30000-90012
Error Messages 30000-90015
Index to MPE 30000-90045
Utilities
EDIT 03000-90012
FCOPY 03000-90064
SORT 32214-90001
Compiler Library 30000-90028
Data Management
Using HP V/3000 32209-90004
HP V/3000 Reference Manual 32209-90001
HP V/3000 Operator's Guide 32209-90003
KSAM/3000 30000-90079
IMAGE/3000 32215-90003
QUERY/3000 30000-90042
Languages
Using COBOL/3000 32213-90003
RPG/3000 Listing Analyzer 32104-90003
FORTRAN/3000 Pocket Guide 32102-90002
BASIC/3000 Pocket Guide 03000-90050
BASIC/3000 Compiler 32103-90001
SPL Pocket Guide 32100-90001
APL/3000 Pocket Guide 32105-90003
COBOL II/3000 32233-90001
RPG/3000 32104-90001
FORTRAN/3000 30000-90040
BASIC/3000 Interpreter 30000-90026
BASIC/3000 for Beginners 03000-90025
SPL 30000-90024
APL/3000 32105-90002
COBOL/3000 to COBOL II/3000 Conversion Guide 32233-90005
Scientific Library 30000-90027
SPL Textbook 30000-90025
Data Communi-cations
Guidebook to Data Communications 5955-1715
Data Communications Handbook 30000-90105
RJE/3000 30000-90047
DS/3000 32190-90001
DS/3000 to DS/1000 32190-90005
MRJE/3000 32192-90001
MTS/3000 32193-90002
Manufacturing
Master Prod Scheduling and Rough Cut Resource Plan 32260-96001
Maintaining Parts and Bills of Materials 32260-90002
Maintaining Routings and Workcenters 32260-90003
Material Issues and Receipts 32260-90004
Maintaining Work Orders 32260-90005
Managing Inventory Balances 32260-90006
Maintaining Purchase Orders 32260-90007
Material Requirements Planning 32260-90008
Standard Product Costing 32260-90009
System Customization 32260-90010
System Operation 32260-90011
Conversions
System/3 to HP 3000 Conversion 32104-90004
3000CX to Series II Conversion 30000-90046
COBOL/3000 to COBOL II/3000 Conversion Guide 32233-90005

Appendix B: MPE III Commands

HEWLETT PACKARD

All MPE III commands are summarized below, grouped alphabetically within capability. The last column denotes when the command can be issued: during a batch job (J), during a session (S), during a break (B), or programmatically (P), through the COMMAND intrinsic.

STANDARD CAPABILITY COMMANDS

Command	Function	When issued
:( ) command log on	Begins a session, executes the enclosed MPE command, and ends the session upon completion of the command.	S
:ABORT	Aborts the current program.	B
:ALTLOG	Alters the attributes of an existing logging identifier	J,S,B,P
:ALTSEC	Changes security provisions for a file	J,S,B,P
:APL	Accesses the APL subsystem.	J,S
:ASSOCIATE	Gives a user operator control of a device	J,S,B,P
:BASIC	Calls BASIC/3000 interpreter.	J,S
:BASICGO	Compiles, prepares, and executes a BASIC/3000 program.	J,S
:BASICOMP	Compiles a BASIC/3000 program.	J,S
:BASICPREP	Compiles and prepares a BASIC/3000 program.	J,S
:BUILD	Creates a new file.	J,S,B,P
:BYE	Terminates a session.	S
:COBOL	Compiles a COBOL/3000 program.	J,S
:COBOLGO	Compiles, prepares, and executes a COBOL/3000 program.	J,S
:COBOLPREP	Compiles and prepares a COBOL/3000 program.	J,S
:COMMENT	Inserts comment into command stream.	J,S,B,P
:CONTINUE	Disregards job-error condition.	J
:DATA	Defines data from outside standard input stream. Cannot be read on $STDINX file. Acceptable for device recognition.	J,S
:DEBUG	Invokes the MPE debug facility.	S,P
:DISASSOCIATE	Removes the control of a device from a user	J,S,B,P
:DISMOUNT	Causes a volume set that was mounted by the user to be dismounted.	J,S,B
:DSLINE	Opens or closes communication line with DS/3000.	J,S
:DSTAT	Displays the current status of the disc drives on the system.	J,S,B,P
:EDITOR	Calls the EDITOR.	J,S
:ELSE	Provides an alternate execution sequence for an IF statement.	J,S,B
:ENDIF	Terminates an IF block.	J,S,B
:EOD	Denotes end of data. Cannot be read on $STDINX file.	J,S
:EOF	Simulates hardware end-of-file on input stream from any device	J,S
:EOJ	Denotes end of batch job. Cannot be read on $STDINX file.	J
:FILE	Defines or redefines a file's characteristics.	J,S,B,P
:FORTGO	Compiles, prepares, and executes a FORTRAN/3000 program.	J,S
:FORTPREP	Compiles and prepares FORTRAN/3000 program.	J,S
:FORTRAN	Compiles a FORTRAN program.	J,S
:FREERIN	Deallocates a global RIN, and returns it to RIN pool.	J,S
:GETLOG	Establishes a logging identifier on the system	J,S,B,P
:GETRIN	Acquires a global RIN.	J,S,B,P
:HELLO	Initiates a session. Acceptable for device recognition. Requires 1A capability class.	S
:HELP	Access the HELP subsystem.	J,S,B
:IF	Used to control the execution sequence of a job.	J,S,B

Command	Function	When issued
:IML	Initiate IML/3000's Inquiry and Development Facility (IDF) Subsystem	
:JOB	Initiates a batch job. Requires BA capability class.	J,S
:LISTF	Lists descriptions of files.	J,S,B,P
:LISTLOG	Lists active logging identifiers	J,S,B,P
:LISTVS	Produces a formatted listing of volume set definition information.	J,S,B,P
:MOUNT	Requests the console operator to mount a volume set.	J,S,B
:MRJE	Initiates execution of the Multi-leaving Remote Job Entry (MRJE) facility.	J,S
:PREP	Prepares a compiled program into segmented form.	J,S
:PREPRUN	Prepares and executes a program.	J,S
:PTAPE	Reads a paper tape without X-OFF control.	S,B,P
:PURGE	Deletes a file from the system.	J,S,B,P
:RECALL	Displays all pending console REPLY messages	J,S,B,P
:REDO	Allows the user to edit a command entry.	S,B
:RELEASE	Temporarily suspends all security provisions for a file.	J,S,B,P
:RELLOG	Removes a logging identifier from the system	J,S,B,P
:REMOTE	Establishes communication between a local computer and a remote computer.	S
:RENAME	Renames a file.	J,S,B,P
:REPORT	Displays total accounting information for a log-on group.	J,S,B,P
:RESET	Resets a formal file designator.	J,S,B,P
:RESETDUMP	Disables the MPE stackdump facility.	J,S,B,P
:RESTORE	Restores a complete fileset, stored off-file.	J,S,B,P
:RESUME	Resumes an interrupted program.	B,S
:RJE	Calls the 2780/3780 Emulator.	J,S
:RPG	Compiles an RPG/3000 program.	J,S
:RPGGO	Compiles, prepares, and executes an RPG/3000 program.	J,S
:RPGPREP	Compiles and prepares an RPG/3000 program.	J,S
:RUN	Loads and executes a program.	J,S
:SAVE	Changes a file to permanent status. Requires SF capability for saving files.	J,S,B,P
:SECURE	Restores suspended security provisions for a file.	J,S,B,P
:SEGMENTER	Calls MPE Segmenter.	J,S
:SETCATALOG	Causes the command interpreter to search a catalog of user-defined commands and to establish a directory entry for each command in the catalog	J,S,B
:SETDUMP	Enables the MPE stackdump facility on abort.	J,S,B,P
:SETJCW	Scans the JCW table for a specified JCW name and updates the value of this JCW.	J,S,B,P
:SETMSG	Disables or enables receipt of user or operator messages at standard list device.	J,S,B,P
:SHOWCATALOG	Lists user-defined command (UDC) files.	J,S,B
:SHOWDEV	Reports status of input/output devices.	J,S,B,P
:SHOWIN	Reports status of input device files.	J,S,B,P
:SHOWJCW	Displays the current state of a job control word.	J,S,B,P
:SHOWJOB	Displays job/session status.	J,S,B,P
:SHOWLOG-STATUS	Displays status information about currently opened log files	J,S,B,P
:SHOWME	Reports job/session status.	J,S,B,P
:SHOWOUT	Reports status of output device files.	J,S,B,P
:SHOWTIME	Displays current date and time-of-day.	J,S,B,P

Command	Function	When issued
:SPEED	Changes input speed or output speed of terminal.	S,B,P
:SPL	Compiles an SPL/3000 program.	J,S
:SPLGO	Compiles, prepares, and executes an SPL/3000 program.	J,S
:SPLPREP	Compiles and prepares an SPL/3000 program.	J,S
:STORE	Stores a set of files off-line.	J,S,B,P
:STREAM	Spools batch jobs or data in session or job mode.	J,S,B,P
:TELL	Transmits a message.	J,S,B,P
:TELLOP	Transmits a message from the user to the computer operator.	J,S,B,P
:VINIT	Accesses the VINIT subsystem to perform on-line conditioning and formatting of serial discs and private volumes.	J,S
:VSUSER	Prints a listing of all users of a currently mounted volume set.	J,S,B

STANDARD CAPABILITY COMMANDS (Segmenter Commands)

Command	Function	When issued
-ADDRL	Adds a procedure to an RL.	J,S
-ADDSL	Adds a segment to an SL.	J,S
-ADJUSTUSLF	Adjusts directory space in a user subprogram library (USL) file.	J,S
-AUXUSL	Designates a source of RBM input for COPY command.	J,S
-BUILDRL	Creates a permanent, formatted RL file.	J,S
-BUILDSL	Creates a permanent, formatted SL file.	J,S
-BUILDUSL	Creates a temporary, formatted USL file.	J,S
-CEASE	Deactivates one or more entrypoints in a USL.	J,S
-CLEANSL	Copies the currently managed SL to a new SL file, removing inactive segments	J,S
-CLEANUSL	Copies the currently managed USL to a new USL file, removing inactive segments	J,S
-COPY	Copies an RBM or segment from one USL to another.	J,S
-COPYSL	Same as CLEANSL except allows user to expand SL space by a given percentage	J,S
-COPYUSL	Same as CLEANUSL except allows user to expand USL space by a given percentage	J,S
-EXIT	Exits from Segmenter, returning control to MPE command interpreter.	J,S
-EXPANDUSLF	Changes length of a USL file.	J,S
-HIDE	Sets an RBM internal flag on.	J,S
-INITUSLF	Initializes buffer for a USL file to the empty state.	J,S
-LISTRL	Lists the procedures in an RL.	J,S
-LISTSL	List specific segments in the SL	J,S
-LISTUSL	List specific segments in USL.	J,S
-NEWSEG	Changes the segment name of an RBM.	J,S
-PREPARE	Prepares RBMS from a USL into a program file.	J,S
-PURGERBM	Deletes one or more RBMs from a USL.	J,S
-PURGERL	Deletes an entrypoint or a procedure from an RL.	J,S
-PURGESL	Deletes an entrypoint or a segment from an SL.	J,S
-REVEAL	Sets an RBM internal flag off.	J,S
-RL	Designates an RL for management.	J,S
-SL	Designates an SL for management.	J,S
-USE	Activates one or more RBM entrypoints.	J,S
-USL	Designates a USL for management.	J,S

SYSTEM MANAGER CAPABILITY COMMANDS

Command	Function	When issued
:ALTACCT	Changes an account's characteristics.	J,S,B,P
:ALTVSET	Modifies volume set definitions.	J,S,B,P
:LISTACCT	Lists attributes of an account.	J,S,B,P
:NEWACCT	Creates a new account.	J,S,B,P
:NEWVSET	Defines private volume sets and classes.	J,S,B,P
:PURGEACCT	Removes an account and users from the system's or the volume set's directory.	J,S,B,P
:PURGEVSET	Deletes an existing volume set.	J,S,B,P
:REPORT	Displays an account's resource usage.	J,S,B,P
:RESETACCT	Resets resource-use counters for an account and its groups.	J,S,B,P

ACCOUNT MANAGER CAPABILITY COMMANDS

Command	Function	When issued
:ALTGROUP	Changes a group's attributes.	J,S,B,P
:ALTUSER	Changes a user's attributes.	J,S,B,P
:LISTACCT	Lists attributes of user's log-on account.	J,S,B,P
:LISTGROUP	Lists attributes of a group in user's log-on account.	J,S,B,P
:LISTUSER	Lists attributes of a user in log-on account.	J,S,B,P
:NEWGROUP	Creates a new group in log-on account.	J,S,B,P
:NEWUSER	Creates a new user in log-on account.	J,S,B,P
:PURGEGROUP	Removes a group from the system's or the volume set's directory.	J,S,B,P
:PURGEUSER	Deletes a user from log-on account.	J,S,B,P
:REPORT	Displays resource-usage counts for log-on account and its groups.	J,S,B,P

SYSTEM SUPERVISOR CAPABILITY COMMANDS

Command	Function	When issued
:ALLOCATE	Permanently allocates a program or procedure in virtual memory.	J,S
:DEALLOCATE	Removes a program or procedure from virtual memory.	J,S,P
:JOBPRI	Sets or changes the priority for batch jobs or sessions.	J,S,B,P
:QUANTUM	Changes a circular subqueue time-quantum.	J,S,B,P
:RESTORE	Returns files to the system.	J,S,B,P
:RESUMELOG	Resumes logging following suspension caused by an error.	J,S,B,P
:SHOWLOG	Displays log file status.	J,S,B,P
:SHOWQ	Displays scheduling subqueue information.	J,S,B,P
:STORE	Stores disc files onto magnetic tape or serial disc.	J,S,B,P
:SWITCHLOG	Closes the current log file, and creates and opens a new log file.	J,S,B,P
:SYSDUMP	Starts configurator dialog and copies MPE to magnetic tape or serial disc.	J,S

CONSOLE OPERATOR COMMANDS

Command	Function
:ABORTIO	Aborts pending I/O requests for a device.
:ABORTJOB	Aborts a job or session.
:ACCEPT	Permits a device to accept job/ sessions and/or data.
:ALLOW	Grants a user access to a specific operator command
:ALTJOB	Alters attributes of waiting job or session.
:ALTSPOOLFILE	Alters attributes of output spooling files.
:BREAKJOB	Suspends an executing job.

CONSOLE OPERATOR COMMANDS

Command	Function
:CONSOLE	Changes the system console from its current device to another job-accepting (non-DS) terminal
:DELETESPOOL-FILE	Deletes a spooled device file.
:DISALLOW	Prohibits a user access to a specified operator command.
:DOWN	Removes a device from normal system use.
:DOWNLOAD	Downloads information to an output device.
=DSLINE	Enables or disables the data communications link under control of the DS/3000 subsystem.
:GIVE	Assigns a DOWNed device to the diagnostics.
:HEADOFF	Stops header/trailer output to a device.
:HEADON	Resumes header/trailer output to a device.
:JOBFENCE	Defines input priorities.
:JOBSECURITY	Controls the availability of certain job commands to a user.
:LDISMOUNT	Logically dismounts a private volume set/class (UV capability required).
:LIMIT	Limits the number of concurrently running jobs/sessions.
:LMOUNT	Logically mounts a private volume/ class on a non-system domain disc drive.
:LOG	Starts, restarts, stops User Logging (LG capability required).
=LOGOFF	Aborts all jobs/sessions and prevents further log-ons by all except HIPRI jobs/sessions.
=LOGON	Enables job/session processing following a LOGOFF command.
=MPLINE	Enables or disables the data communications link under control of the MTS/3000 subsystem.
=MRJE	Enables or disables the data communications link under control of the MRJE/3000 subsystem.
:OUTFENCE	Defines priorities for output spooled files.
:REFUSE	Disallows jobs/sessions and/or data on a designated device.
:REPLY	Replies to a pending console request.
=REPLY	Same as :REPLY
:RESUMEJOB	Resumes a suspended job.
:RESUMESPOOL	Resumes a spooled device.
=SHUTDOWN	Closes down the operating system.
:STARTSPOOL	Initiates spooling of a device.
:STOPSPOOL	Terminates spooling of a device.
:STREAMS	Enables or disables the users' ability to submit job/session and/or data streams.
:SUSPENDSPOOL	Causes a spooled device to stop operation.
:TAKE	De-assigns a device that was GIVEn to the diagnostics.
:UP	Allows a DOWNed device to function again.
:VMOUNT	Enables or disables the private volumes facility.
:WARN	Sends an urgent message to jobs and sessions.
:WELCOME	Defines the message users receive when they log on the system.

Appendix C: MPE III Intrinsics

HEWLETT PACKARD

All MPE III intrinsics (system procedures) are summarized below, listed alphabetically. Any special capabilities required to call a particular intrinsic are noted.

Intrinsic	Function
ACCEPT	Accepts (and completes) the request received by the preceding GET intrinsic call.
ACTIVATE	Activates a process. (Requires PH capability.)
ADJUSTUSLF	Adjust directory space in a USL file.
ALTDSEG	Changes the size of an extra data segment. (Requires DS capability.)
ARITRAP	Enables or disables internal interrupt signals from all hardware arithmetic traps.
ASCII	Converts a number from binary to ASCII code.
BINARY	Converts a number from ASCII to binary code.
CALENDAR	Returns the calendar date.
CAUSEBREAK	Requests a session break.
CLOCK	Returns the time of day.
CLOSELOG	Closes access to the logging facility.
COMMAND	Executes an MPE command programmatically.
CREATE	Creates a process. (Requires PH capability.)
CTRANSLATE	Converts a string of characters from EBCDIC to ASCII or from ASCII to EBCDIC.
DASCII	Converts a value from double-word binary to ASCII code.
DBINARY	Converts a number from ASCII to double-word binary value.
DEBUG	Sets breakpoints and modifies or displays stack or register contents.
DLSIZE	Changes size of DL-DB area.
DMOVIN	Copies block from data segment to stack. (Requires DS capability.)
DMOVOUT	Copies block from stack to data segment. (Requires DS capability.)
EXPANDUSLF	Changes length of a USL file.
FATHER	Requests PIN of father process. (Requires PH capability.)
FCARD	Drives the HP 7260A Optional Mark Reader.
FCHECK	Requests details about file input/output errors.
FCLOSE	Closes a file.
FCONTROL	Performs various control operations on a file or terminal device.
FERRMSG	Returns message corresponding to FCHECK error number.
FGETINFO	Requests access and status information about a file.
FINDJCW	Searches the job control word table for a specified job control word (JCW).
FLOCK	Dynamically locks a file.
FMTCALENDAR	Converts the calendar date obtained with the CALENDAR intrinsic.
FMTCLOCK	Converts the time of day obtained with the CLOCK intrinsic.
FMTDATE	Converts calendar date and time of day obtained with the CALENDAR and CLOCK intrinsics.
FOPEN	Opens a file.
FPOINT	Resets the logical record pointer for a sequential disc file.
FREAD	Reads a logical record from a sequential file.
FREADDIR	Reads a logical record from a direct-access disc file.
FREADLABEL	Reads a user file label.
FREADSEEK	Prepares, in advance, for reading from a direct-access file.
FREEDSEG	Releases an extra data segment. (Requires DS capability.)
FREELOCRIN	Frees all local RINs from allocation to a job.
FRELATE	Declares a file pair interactive or duplicative.
FRENAME	Renames a disc file.
FSETMODE	Activates or deactivates file access modes.
FSPACE	Spaces forward or backward on a file.
FUNLOCK	Dynamically unlocks a file.
FUPDATE	Updates a logical record residing in a disc file.

Intrinsic	Function
FWRITE	Writes a logical record to a sequential file.
FWRITEDIR	Writes a logical record to a direct-access disc file.
FWRITELABEL	Writes a user file label.
GENMESSAGE	Accesses the message system.
GET	Receives the next request from the remote master program.
GETDSEG	Creates an extra data segment. (Requires DS capability.)
GETJCW	Fetches contents of job control word.
GETLOCRIN	Acquires a local RIN.
GETORIGIN	Determines source of process activation call. (Requires PH capability.)
GETPRIORITY	Reschedules a process. (Requires PH capability.)
GETPRIVMODE	Dynamically enters privileged mode. (Requires PM capability.)
GETPROCID	Requests PIN of a son process. (Requires PH capability.)
GETPROCINFO	Requests status information about a father or son process. (Requires PH capability.)
GETUSERMODE	Dynamically returns to non-privileged mode. (Requires PM capability.)
INITUSLF	Initializes a buffer for a USL file to the empty state.
IODONTWAIT	Initiates completion operations for an I/O request.
IOWAIT	Initiates completion operations for an I/O request.
KILL	Deletes a process. (Requires PH capability.)
LOADPROC	Dynamically loads a library procedure.
LOCKGLORIN	Locks a global RIN.
LOCKLOCRIN	Locks a local RIN.
MAIL	Tests mailbox status. (Requires PH capability.)
MYCOMMAND	Parses (delineates and defines parameters) for user-supplied command image.
PAUSE	Suspends the calling process for a specified number of seconds.

Intrinsic	Function
PCHECK	Returns an integer code specifying the completion status of the most recently executed slave program-to-program intrinsic.
PCLOSE	Terminates the remote slave program's process.
PCONTROL	Transmits a tag field to the remote slave program and receives a tag field back from the slave.
POPEN	Initiates and activates a slave process in a remote HP 3000 and initiates program-to-program communication with the slave program.
PREAD	Sends a read request to the remote slave program asking the slave to send a block of data back to the master.
PRINT	Prints character string on job/session list device.
PRINTFILEINFO	Prints a file information display on the job/session list device.
PRINTOP	Prints a character string on operator's console.
PRINTOPREPLY	Prints a character string on the operator's console and solicits a reply.
PROCTIME	Returns a process's accumulated central-processor time.
PTAPE	Accepts input from tapes not containing X-OFF control characters.
PUTJCW	Puts the value of a particular job control word (JCW) in the JCW table.
PWRITE	Sends a block of data to the remote slave program.
OPENLOG	Provides access to a logging facility.
QUIT	Aborts a process.
QUITPROG	Aborts a program.
READ	Reads an ASCII string from an input device.
READX	Reads an ASCII string from an input device.
RECEIVEMAIL	Receives mail from another process. (Requires PH capability.)
REJECT	Rejects a request received by the preceding GET intrinsic call and returns an optional tag field back to a remote master program.

Intrinsic	Function
RESETCONTROL	Resets a terminal to accept a CONTROL-Y signal.
RESETDUMP	Disables the abort stack analysis facility.
SEARCH	Searches an array for a specified entry or name.
SENDMAIL	Sends mail to another process. (Requires PH capability.)
SETDUMP	Enables the stack analysis facility.
SETJCW	Sets bits in job control word.
STACKDUMP	Dumps selected parts of stack to a file.
SUSPEND	Suspends a process. (Requires PH capability.)
SWITCHDB	Switches DB register pointer. (Requires PM capability.)
TERMINATE	Terminates a process.
TIMER	Returns system-timer bit-count.
UNLOADPROC	Dynamically unloads a library procedure.
UNLOCKGLORIN	Unlocks a global RIN.
UNLOCKLOCRIN	Unlocks a local RIN.
WHO	Returns user attributes.
WRITELOG	Writes a record to a logging file.
XARITRAP	Arms the software arithmetic trap.
XCONTRAP	Arms or disarms the CONTROL-Y trap.
XLIBTRAP	Arms or disarms the software library trap.
XSYSTRAP	Arms or disarms the sytem trap.
ZSIZE	Changes size of Z-DB area.

Appendix D: HP 3000 Series 30 & Series 33 Hardware Features

HEWLETT PACKARD

This appendix summarizes the hardware features of the HP 3000 Series 30 and Series 33. To learn more about the HP 3000 architecture, refer to Chapter 1 System Introduction and Chapter 6 System Architecture.

The HP 3000 Series 30 and Series 33 are designed around independent elements that are organized together in a central bus structure. The elements of the system consist of a central processor that operates through a bus interface controller, memory arrays with a memory controller, general I/O channels, asynchronous data communications controllers, and a bus system that enables communications between the I/O elements. Also, the system includes a system console, system front panel, and a maintenance facility. Peripheral elements attach to the system through the general I/O channels. Interactive terminals attach to the system through the asynchronous data communications controllers.

The CPU is controlled by a specially designed Hewlett-Packard silicon-on-sapphire (SOS) microprocessor to allow a great deal of flexibility in the machine instruction set. The HP 3000 also employs high-speed, semiconductor memory modules which use automatic fault detection and correction.

The design of the HP 3000 Series 30 and Series 33 hardware provides an efficient and powerful foundation upon which the software is built, as illustrated in Figure D-1. The configuration of hardware modules and peripheral devices is easily changed to accommodate system expansion.

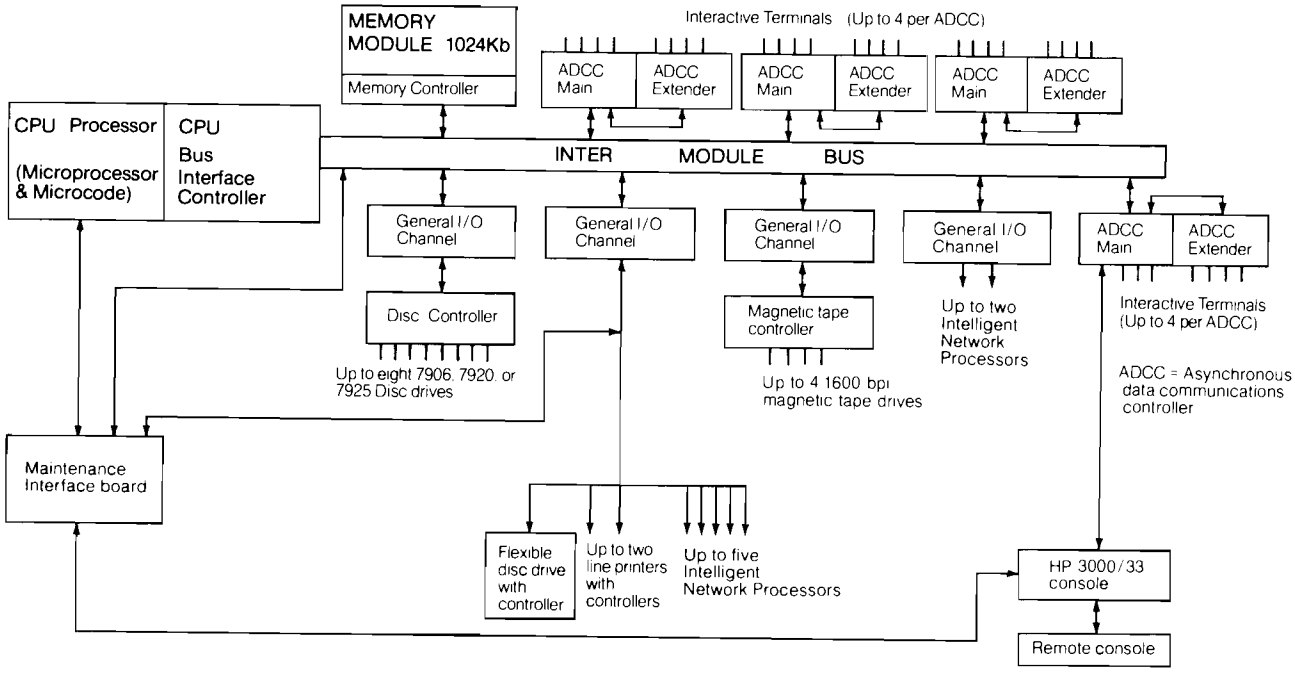


Figure D-1. HP 3000 SERIES 33 HARDWARE ORGANIZATION (MAXIMUM CONFIGURATION)

Central processing unit (CPU)

The significant features of the HP 3000 Series 30 and Series 33 central processing unit (CPU) are listed in Table D-1.

Table D-1. HP 3000 CPU Features

Architecture

- Hardware-implemented stack
- Separation of code and data
- Non-modifiable, re-entrant code
- Variable-length code segmentation
- Virtual memory for code
- Dynamic relocatability of programs

Implementation

- Microprogrammed SOS/CMOS CPU
- 90 nanosecond cycle time; microinstructions execute in 5 to 7 cycles (4.5 cycle average)
- Automatic restart after power failure
- Inter-module bus
- Overlapping CPU and I/O operations

Instructions

- 214 powerful instructions
- Instructions are 8, 16, or 32 bits in length
- 16- and 32-bit integer arithmetic
- 32- and 64-bit floating point arithmetic
- 28-digit packed decimal arithmetic
- Special instructions that optimize the efficiency of the operating system.

The CPU converts an instruction in the current instruction register (CIR) into a starting address for the microcode contained in a read-only memory (ROM), and determines various initial conditions required for executing the instruction. As the current instruction is being executed, the next instruction is fetched and placed into the next instruction register (NIR). Upon completion of the current instruction, the contents of NIR are loaded into CIR and the cycle is repeated. This "pipelining" of the current instruction execution with the next instruction-fetch improves throughput by overlapping operations. The HP 3000 Series 30 and Series 33 instruction set is presented in Appendix E. Instructions are 16 or 32 bits in length except stack operations, which are 8-bit instructions. These include a variety of memory reference, branch, arithmetic and data manipulation instructions that operate on integer, real, logical, packed decimal, character and string data. Floating point arithmetic can be performed in single precision (32 bits) or double precision (64 bits), integer arithmetic in 16-bit and 32-bit lengths, and packed decimal instructions extend to 28 digits of precision. In addition, there are a number of instructions designed to aid in creating the multiprogramming environment of the system. These include procedure call and exit instructions and others which implement various operating system functions previously done in software.

Firmware storage and control consists of microcode, stored in read-only memory (ROM), and associated logic control. Microcode routines control the operation of the instruction decoder and the hardware processor, in order to create the HP 3000 operating environment. The control storage has a cycle time of 90 nanoseconds and a microinstruction execution time which is variable from 270 to 630 nanoseconds.

The hardware processor consists of an arithmetic-logic unit, shifting network, 27 specific purpose registers—13 of which are accessible to user programs—and related data manipulating and testing logic. Since the HP 3000 architecture (see Chapter 6) is structured on code segments and data segments, most of the CPU registers are used for defining the segment limits and operating elements within the segments. As shown in Figure D-2, three of the CPU registers point to locations in a code segment defined as the current code segment. Six of the registers point to locations in a data segment defined as the current data segment. Table D-2 lists all 27 registers and their associated functions.

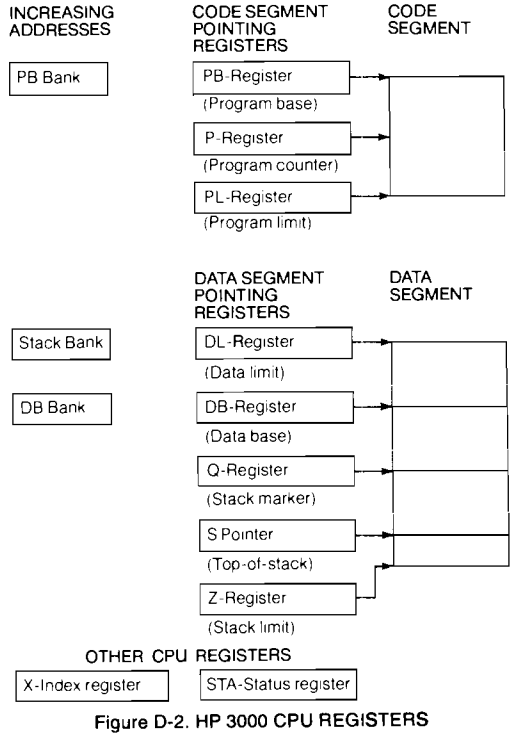


Figure D-2. HP 3000 CPU REGISTERS

HEWLETT PACKARD

Table D-2 Series 30 and Series 33 Hardware Registers

Registers accessible to user programmers

Register	Function
PB	Code Segment Pointers
P	
PL	
DL	Stack Pointers
DB	
Q	
SB	
S	
DST	
Z	
TOSA	Top of Stack Registers
TOSB	
X	Index Register
STA	Status Register

Registers dedicated for system use

CIR	Current Instruction Register
NIR	Next Instruction Register
SP0	Scratch Pad, Flag, and Interrupt Registers
SP1	
SP2	
SP3	
SP4	
SP5	
ISR	
CNTR	
MEMA	Memory Address and Data Registers
BUSD	
RAR	Firmware Address Register

The two top of stack registers are of special interest. In order to improve execution speed, up to two elements from the top of a data stack may be contained in these registers. This allows many functions to be treated as register-to-register operations rather than the slower speed memory-to-register or register-to-memory type operations. These registers are manipulated by the CPU, and their use is fully transparent.

Main memory

The significant features of the HP 3000 Series 30 and Series 33 main memory are listed in Table D-3.

Table D-3: HP 3000 Series 30 and Series 33 Main Memory Features

- High-speed, semiconductor, random access memory
- Automatic fault detection and fault correction
- Memory sizes ranging from 256k bytes to 1 megabyte
- Write: 860 nsec minimum cycle time
- Read: 430 nsec access, 860 nsec cycle time
- 30 to 45 minute rechargable battery packs to maintain memory data during power failure. Total amount of time depends on battery condition, age, and level of charge

The HP 3000 Series 30 and Series 33 use high-speed, semiconductor, random access memory (16k MOS RAM) which provides single-bit correction and all double bit fault detection.

Due to the modular design of the HP 3000, any system can be easily expanded from one memory size to another.

The word length transmitted over the bus is 16 bits. In the memory modules the word length is expanded to 22 bits; 16 bits of data and six bits for automatic fault detection and correction.

All detected memory faults are automatically logged to special (non-user) storage. MPE periodically reads this storage and writes the information to the disc file. This file is accessed by an HP Customer Engineer, from a terminal on the system, while performing preventative maintenance. If memory chips have a history of failures, they are replaced during maintenance.

Operating power for the memory modules is supplied by a rechargeable battery pack in the semiconductor memory power supply. When the power supply input voltage is removed, battery power is available for up to 4 hours (depending on memory size and battery condition) to maintain memory data.

Communication between the CPU, memory and I/O modules is carried over the inter-module bus (IMB). Because the CPU generates greater than 90 percent of the bus activity, it is given continuous access to the bus and relinquishes control to the I/O channels only on request.

The IMB has separate address and data paths, each with handshake controls that operate in a master/slave mode to transfer data between modules. The CPU talks to memory and to the I/O system and always functions as a master. The I/O channels function as masters to memory but become slaves when talking with the CPU. To access memory, the I/O channels must request the bus through a priority structure. Any channel request will cause the CPU to relinquish control of the IMB so that the request can be serviced.

Input/output

All access to input/output devices is by way of the device-independent MPE file system. All location of data, buffering, data transfers, and deblocking are handled automatically by MPE. When you ask to read a named file, you are only implicitly specifying the actual disc address of the file; the file system determines the explicit address and performs the read. At another level, when you ask the file system for a certain type of device by specifying a device class name (e.g., magnetic tape, line printer, etc.), the file system takes care of allocating an actual device. If you must have actual contact with specific devices, you may address them directly. Below this single, flexible interface is a powerful and carefully balanced hardware/software input/output system.

All devices can be operated concurrently (within system bandwidth). Peripherals that fail are taken off-line from the operating system by operator command.

Peripheral I/O hardware

HP 3000 Series 30 and Series 33 peripheral I/O hardware consists of the general I/O channels (GIC), Hewlett-Packard Interface Bus (HP-IB), device controllers, and the peripheral units. Asynchronous data communications controllers interface log-on and data-entry terminals. Table D-4 lists the features which this peripheral I/O hardware offers.

Table D-4: HP 3000 Peripheral I/O Hardware Features

- 4 ports per Asynchronous Data Communications Controller
- Up to 8 terminal controllers per system
- Type 103 and 212 modem support

When an I/O request is issued, the device driver in the CPU assembles the channel program, then issues a Start I/O Program (SIOP) instruction to one of two types of channels on the Inter-Module Bus: the General I/O Channel (GIC) and the Asynchronous Data Communications Controller (ADCC). The GIC is the hardware I/O channel which provides the electrical interface between the computer system via the IMB and peripheral devices connected to the Hewlett-Packard Interface Bus (HP-IB). The HP-IB is HP's implementation of the IEEE standard 488-1975 interface, used on the Series 30 and Series 33 to connect peripheral devices to the channel. The HP-IB consists of eight data lines and eight control lines. The ADCC provides a bit-serial data interface between the computer system and terminals. The two channels operate in a similar manner; however, the GIC has a DMA facility to permit high-speed transfer of large blocks of data, while the ADCC can transfer data only one character at a time.

General I/O channel

The General I/O Channel is the primary channel for communication between the CPU and the I/O devices other than terminals. Each GIC controls a Hewlett-Packard Interface Bus (HP-IB) and translates I/O commands from the CPU into the proper HP-IB protocol. Nearly all transactions with I/O devices are accomplished without software interrupts, since I/O is achieved with channel programs. Software is responsible for setting up a channel program, but the execution of this program is performed by the CPU's channel microcode. The CPU's channel microcode is devoted to I/O tasks and implements the necessary algorithms for decoding the channel instructions and effecting the required I/O operations. Once the channel program is running, device control and data flow are normally carried to completion with no software intervention and without altering the system environment. If special situations arise, software may alter the program or even halt execution.

Several devices may simultaneously need service, and the CPU must decide which one will receive attention. First, all channels are polled, and the highest priority GIC with a device request pending is chosen. The CPU then obtains from that channel the number of the highest priority device needing service. Once the device number is determined, execution of the channel program will begin. The CPU fetches each channel instruction and breaks it down into several IMB commands addressed to the proper GIC. The GIC interprets these commands and directs them onto the HP-IB device.

The GIC contains DMA (Direct Memory Access) hardware which allows large records of data to be transferred at the maximum speed of the HP-IB (about 1 Mb/sec.). The channel microcode enables the device and then initializes the DMA hardware on the GIC. After initial addressing of a device to talk or listen, the CPU relinquishes control of the IMB and allows the GIC to perform its function through DMA operation. During this time the GIC becomes the master of the bus and memory and controls traffic flow. On a read operation the DMA hardware will read the bytes, pack them into words and place them directly into memory, all without assistance from the CPU. The CPU is free to service other devices while DMA is in progress. Upon completion of a DMA transfer, the GIC returns to a slave condition and awaits the next operation.

Asynchronous data communications controller

The Asynchronous Data Communications Controller (ADCC) is the second channel type used in the system. This channel performs for terminals essentially the same functions as the GIC but not in the same manner. Data is transferred from memory to the ADCC in parallel form, then converted to a serial bit stream for transmission over the RS-232-C lines to the devices. Information being read from a device is in serial form and is converted to eight-bit bytes for transfer to memory.

Two types of ADCC boards may be used, the Main ADCC and the Extender ADCC. Each board contains four ports for connection to devices through RS-232-C data communications lines. The Main ADCC supports full duplex (103, 212, and 202T type modems in the U.S.) only; Extender ADCC boards are required for European half duplex modem support. The Main ADCC is used when four or fewer devices are connected to a channel. The Extender ADCC extends the device capability of the channel to eight. Most of the control circuitry is on the Main ADCC. For this reason, the Main ADCC is required for the Extender ADCC to function.

When more than eight devices are to be attached to ADCC channels, additional Main ADCCs are required, since each ADCC can accommodate only one Extender.

Unlike the GIC, the ADCC does not have a DMA facility and therefore cannot be a master of the IMB or of memory. Also, terminals on the channel do not respond to a parallel poll. As a result, the ADCC is always a slave and must be directly controlled by the CPU through the use of channel programs. Circuitry on the ADCC decodes address information relating to channel and devices, and selects the correct device for operation.

The ports on the ADCC (Main and Extender) may be either hardwired to devices or to modems.

Device controller

The device controller is the hardware linkage between a peripheral device and the computer system. Its primary function is to translate I/O commands from a general I/O channel to the unique signals required to control a particular device. When an I/O program is in execution, the device controller responds to and requests service from the general I/O channel. The device controller also generates interrupts when required by some device condition or by channel command.

Device reference table (DRT)

Device controllers are identified by a logical device number which is used to access the device reference table (DRT). The DRT is known to both hardware and software containing, among other things, a pointer to the start of the SIO program for each device controller. Each device controller connects to a General I/O Channel (GIC). Certain device controllers may control several logical devices. In such cases, each logical device attached to the controller is addressed separately using a unit number assigned when the device is installed.

Data service and interrupt priorities

In addition to a logical device number, there are two other characteristic numbers associated with each device controller. These are the data service priority and interrupt priority. In the Series 30 and Series 33, both of these are determined by the logical device number in the DRT: the lower the number, the higher the priority.

Interrupt system

The interrupt system provides for up to 105 external interrupt levels. When interrupts occur, the microprogrammed interrupt handler identifies each interrupt and grants control to the highest priority interrupt. Current operational status is saved by the microprogram, which then sets up the interrupt processing environment and transfers control to the interrupt routine.

Interrupt routines operate on a common stack (interrupt control stack) which is known to both hardware and software. This feature permits nesting of interrupt routines in the case of multiple interrupts, thus allowing higher priority devices to interrupt lower priority devices.

The interrupt system also provides for 20 internal interrupts (for user errors, system violations, hardware faults, and power fail/restart) plus fourteen traps for arithmetic errors and illegal use of instructions.

Peripherals

The peripheral devices used on the HP 3000 Series 30 and Series 33 are connected primarily to GICs, while the ADCC is reserved solely for terminals. Peripherals attached to GICs through the HP-IB include the system disc drive and controller (which can have up to 7 additional drives "daisy-chained" from it), flexible disc drive, line printers, and magnetic tape drives. For a complete configuration of the supported peripherals on the HP 3000 Series 30 and Series 33, you are referred to the current HP 3000 Price/Configuration Guide.

Automatic restart after power failure

An integral part of the HP 3000 Series 30 and Series 33 is a power fail/automatic restart capability. When the system AC line voltage falls below 10% of nominal voltage, the system initiates a powerfail warning (PFW). During PFW the system (hardware and MPE) writes all register contents to a reserved section of main memory, activities in the system are successfully completed, and then the power down signal is generated and the system is shut down. The battery back-up power supply refreshes main memory and ensures its validity for up to 45 minutes, depending on memory size and battery condition.

The system is automatically restarted when all power supply voltages reach 90% of their normal values and all register values are automatically restored and processing resumes.

System Serviceability

The Series 30 and Series 33 are designed to be both extremely reliable and easy to service. The packaging makes all system components as accessible as possible. The power supplies and flexible disc units are mounted on sliding "rails" for easy removal and servicing. All power distribution is through quick release connectors rather than cumbersome screw terminal strips.

A totally new feature that enhances serviceability both in hardware and software is the use of the system console as the maintenance console. Through a new Maintenance Interface board and maintenance mode software loaded through a data cartridge in the system console, HP field personnel will have a complete maintenance display in English and octal values on the system console. Values such as the contents of all registers, dynamically selected memory contents (16 words at a time), and system status displays are available quickly from keystrokes entered on the console keyboard.

The first use of the maintenance console is by the customer: a system self-test is provided with the HP 3000 on a terminal data cartridge. In less than two minutes the diagnostic will check out all hardware components involved in a system "cold-load." Faults are isolated to the module level, with concise, yet easy to understand messages printed on the console CRT display. ***Because of the simplicity and ease of use of this self test, you must run the system self-test prior to calling Hewlett-Packard for hardware maintenance.***

If a service call is necessary, HP Customer Engineers and Operating System Specialists can use the console CRT display to inquire into the status of diagnostics initiated from the console system and even into the status of hardware registers for detailed trouble shooting.

Remote system verification program (RSVP)

Making this new maintenance console even more valuable is the ability to transmit the display and control functions to a remote HP 2645 terminal via a modem and telephone link. With this facility, the CE on site can call the HP Service Office and have a Specialist get "on-line" to the system over the telephone via the remote system console/maintenance console. The CE loads the remote maintenance code data cartridge into the console (15 seconds), then switches the modem (user-supplied) to the console using a switch built into the terminal junction panel to establish the telephone link. The Specialist now has a duplicate display of the HP 3000 system console/maintenance console display, with the ability to send the CE and/or system manager messages that are not transmitted to the computer. This "remote maintenance console" facility is a standard part of all Series 30 and Series 33 systems. You are required to have a Bell 103 type modem (300 baud) or Bell 212 (1200 baud) for use in connecting the console to the phone line. Throughout the procedure, complete control over access to the system remains with you. As a back-up capability to the system console, the system front panel is hardwired to perform console control commands (only) as well.

Diagnostics

Several levels of diagnostic software help identify and diagnose hardware problems in the HP 3000 computer system. The levels of diagnostics are:

- Verification programs for peripherals run under MPE
- Stand-alone diagnostics to verify all system modules
- PC board microdiagnostics in PROM for CPU, Memory, and I/O

Appendix E: HP 3000 Series 30 and Series 33 Machine Instructions

HEWLETT PACKARD

STACK OP INSTRUCTIONS

ADAX	Add A to X
ADBX	Add B to X
ADD	Add A to B
ADXA	Add X to A
ADXB	Add X to B
AND	Logical AND of A and B
BTST	Test byte on TOS and set CC
CAB	Rotate A-B-C
CMP	Integer compare B, A and set CC
DADD	Double integer add D, C + B, A
DCMP	Double integer compare and set CC
DDEL	Double delete TOS
DDIV	Double integer divide
DDUP	Double duplicate TOS
DECA	Decrement A
DECB	Decrement B
DECX	Decrement X
DEL	Delete TOS
DELB	Delete B
DFLT	Float a double integer
DIV	Integer divide B by A
DIVL	Divide long integer C, B ÷ A
DMUL	Double integer multiply
DNEG	Double integer negate
DSUB	Double integer subtract D, C − B, A
DTST	Test double word on TOS and set CC
DUP	Duplicate TOS
DXCH	Double exchange
DZRO	Push double zero onto stack
FADD	Floating point add, D, C + B, A
FCMP	Floating point compare and set CC
FDIV	Floating point divide D, C ÷ B, A
FIXR	Fix and round
FIXT	Fix and truncate
FLT	Float an integer
FMPY	Floating point multiply
FNEG	Floating point negate
FSUB	Floating point subtract D,C—B,A
INCA	Increment A
INCB	Increment B
INCX	Increment X
LADD	Logical add A + B
LCMP	Logical compare B, A and set CC
LDIV	Logical divide C, B ÷ A
LDXA	Load X into A
LDXB	Load X into B
LMPY	Logical multiply B × A
LSUB	Logical subtract B − A
MPY	Multiply integers, integer product
MPYL	Multiply integers, long integer product
NEG	Integer negate
NOP	No operation
NOT	Logical complement TOS
OR	Logical OR of A, B
STAX	Store A into X
STBX	Store B into X
SUB	Integer subtract B − A
TEST	Test TOS and set CC
XAX	Exchange A and X
XBX	Exchange B and X
XCH	Exchange A and B
XOR	Logical exclusive OR of A, B
ZERO	Push integer zero onto stack
ZROB	Zero B
ZROX	Zero X

SHIFT INSTRUCTIONS

ASL	Arithmetic shift left
ASR	Arithmetic shift right
CSL	Circular shift left
CSR	Circular shift right
DASL	Double arithmetic shift left
DASR	Double arithmetic shift right
DCSL	Double circular shift left
DCSR	Double circular shift right
DLSL	Double logical shift left
DLSR	Double logical shift right
LSL	Logical shift left
LSR	Logical shift right
QASL	Quadruple arithmetic shift left
QASR	Quadruple arithmetic shift right
TASL	Triple arithmetic shift left
TASR	Triple arithmetic shift right
TNSL	Triple normalizing shift left

LEGEND

TOS	Top of stack	A	Top of stack	D	Location below C
CC	Condition Code	B	Location below A	DB	Data Base
X	Index Register	C	Location below B	DL	Data Limit

FIELD AND BIT INSTRUCTIONS

DPF	Deposit field, A bits to B
EXF	Extract specified field, right-justify
SCAN	Scan bits
TBC	Test specified bit and set CC
TCBC	Test and complement bit, set CC
TRBC	Test and reset bit, set CC
TSBC	Test and set bit, set CC

BRANCH INSTRUCTIONS

BCC	Branch on specified CC
BCY	Branch on carry
BNCY	Branch on no carry
BNOV	Branch on no overflow
BOV	Branch on overflow
BR	Branch unconditionally
BRE	Branch on TOS even (bit 15 = 0)
BRO	Branch on TOS odd (bit 15 = 1)
CPRB	Compare range and branch
DABZ	Decrement A, branch if zero
DXBZ	Decrement X, branch if zero
IABZ	Increment A, branch if zero
IXBZ	Increment X, branch if zero

MOVE INSTRUCTIONS

CMPB	Compare bytes in two memory blocks
MABS	Move using absolute addresses
MDS	Move using data segments
MFDS	Move from data segment
MOVE	Move words in memory, addresses +/−
MTDS	Move to data segment
MVB	Move bytes in memory, addresses +/−
MVBL	Move words from DB+ to DL+ area
MVBW	Move bytes while of specified type
MVLB	Move words from DL+ to DB+ area
SCU	Scan bytes until test or terminal byte
SCW	Scan bytes while equal to test byte

PRIVILEGED MEMORY REFERENCE INSTRUCTIONS

LDEA	Load double word from extended address
LSEA	Load single word from extended address
LST	Load from system table
PLDA	Privileged load from absolute address
PSTA	Privileged store into absolute address
SDEA	Store double word into extended address
SSEA	Store single word into extended address
SST	Store into system table

IMMEDIATE INSTRUCTIONS

ADDI	Add immediate to integer in A
ADXI	Add immediate to X
ANDI	Logical AND immediate with A
CMPI	Compare A with immediate, set CC
CMPN	Compare A with negative immediate
DIVI	Divide immediate into A
LDI	Load immediate to TOS
LDNI	Load negative immediate to TOS
LDXI	Load X immediate
LDXN	Load X negative immediate
MPYI	Multiply immediate with A
ORI	Logical OR immediate with A
SBXI	Subtract immediate from X
SUBI	Subtract immediate from A
XORI	Logical exclusive OR immediate

REGISTER CONTROL INSTRUCTIONS

ADDS	Add operand to stack pointer
PSHR	Push specified registers onto stack
RCLK	Read clock
SCLK	Store clock
SETR	Set specified registers from stack
SUBS	Subtract operand from stack pointer
XCHD	Exchange DB and TOS

PROGRAM CONTROL AND SPECIAL INSTRUCTIONS

DISP	Dispatch	PCN	Push CPU code (% 10)
EXIT	Exit from procedure	PSDB	Pseudo interrupt disable
HALT	Halt	PSEB	Pseudo interrupt enable
IXIT	Interrupt exit	RSW	Push cold load chan/dev
LLBL	Load label	SCAL	Subroutine call
LLSH	Linked list search	SXIT	Exit from subroutine
LOCK	NOP	UNLK	NOP
PAUS	Pause, interruptable	XEQ	Execute stack word
PCAL	Procedure call		

I/O INSTRUCTIONS

CBKP	Clear breakpoint	SCLR	Set system clock limit
HIOP	Halt I/O program	SED	Set enable/disable external interrupts
INIT	Initialize I/O channel	SMSK	Set device mask
RCCR	Read system clock	TOFF	Hardware timer off
RIOC	Read I/O channel	TON	Hardware timer on
RMSK	Read device mask	WIOC	Write I/O channel
SBKP	Set breakpoint	IOCL	I/O clear
SEML	Semaphore Load	ROCL	Channel roll call
STRT	Programmatic warm start	MCS	Memory controller read status
SIOP	Start I/O channel program	DUMP	Programmatic dump

LOOP CONTROL INSTRUCTIONS

MTBA	Modify variable, test against limit, branch	TBA	Test variable against limit, branch
MTBX	Modify X, test against limit, branch	TBX	Test X against limit, branch

MEMORY ADDRESS INSTRUCTIONS

ADDM	Add memory to TOS	LDX	Load X
CMPM	Compare TOS with memory	LOAD	Load word onto stack
DECM	Decrement memory	LRA	Load relative address onto stack
INCM	Increment memory	MPYM	Multiply TOS by memory
LDB	Load byte onto stack	STB	Store byte on TOS into memory
LDD	Load double word onto stack	STD	Store double on TOS into memory
LDPN	Load double from program, negative	STOR	Store TOS into memory
LDPP	Load double from program, positive	SUBM	Subtract memory from TOS

EXTENDED INSTRUCTION SET

Extended-Precision Floating Point

EADD	ADD
ECMP	Compare
EDIV	Divide
EMPY	Multiply
ENEG	Negate
ESUB	Subtract

Decimal Arithmetic

ADDD	Decimal Add
CMPD	Decimal compare
CVAD	ASCII to decimal conversion
CVBD	Binary to decimal conversion
CVDA	Decimal to ASCII conversion
CVDB	Decimal to binary conversion
DMPY	Double logical multiply
MPYD	Decimal multiply
NSLD	Decimal normalizing left shift
SLD	Decimal left shift
SRD	Decimal right shift
SUBD	Decimal subtract

The design of the HP 3000 Series III hardware provides an efficient and powerful foundation upon which the software is built, as illustrated in Figure F-1. Communication between modules occurs over the central data bus. The central processing unit (CPU) and the input/output processor (IOP), although independent of one another, share a common module address. This is resolved by giving the IOP a higher priority in the case where both processors concurrently request use of the bus. The CPU is controlled by a specially designed microprocessor to allow a great deal of flexibility in the machine instruction set. The HP 3000 also employs high-speed, semiconductor memory modules which use automatic fault detection and correction with no loss in performance.

The basic structure of independent modules organized around a central data bus permits high-speed internal data rates. When not communicating over the bus, each module can run independently at its own speed.

The presence of a separate IOP b[illegible] dedicated to input/output data transfers me[illegible] HP 3000 can always respond immediately to the ne[illegible] I/O devices regardless of what transfers are currently in progress between the various system modules. It also means that many I/O operations can be handled concurrently with CPU, main memory, and high-speed selector channel operations.

Data may be transferred directly between main memory and high-speed peripheral devices in block mode by way of high-speed selector channels connected to the central data bus. For lower speed devices, data may be multiplexed on a word-by-word basis by way of the IOP and a multiplexer channel. In both cases the I/O channels execute in parallel with CPU operations. Direct control of devices attached to the IOP bus is also possible through the use of the CPU's direct I/O instructions.

The configuration of hardware modules and peripheral devices is easily changed to accommodate system expansion.

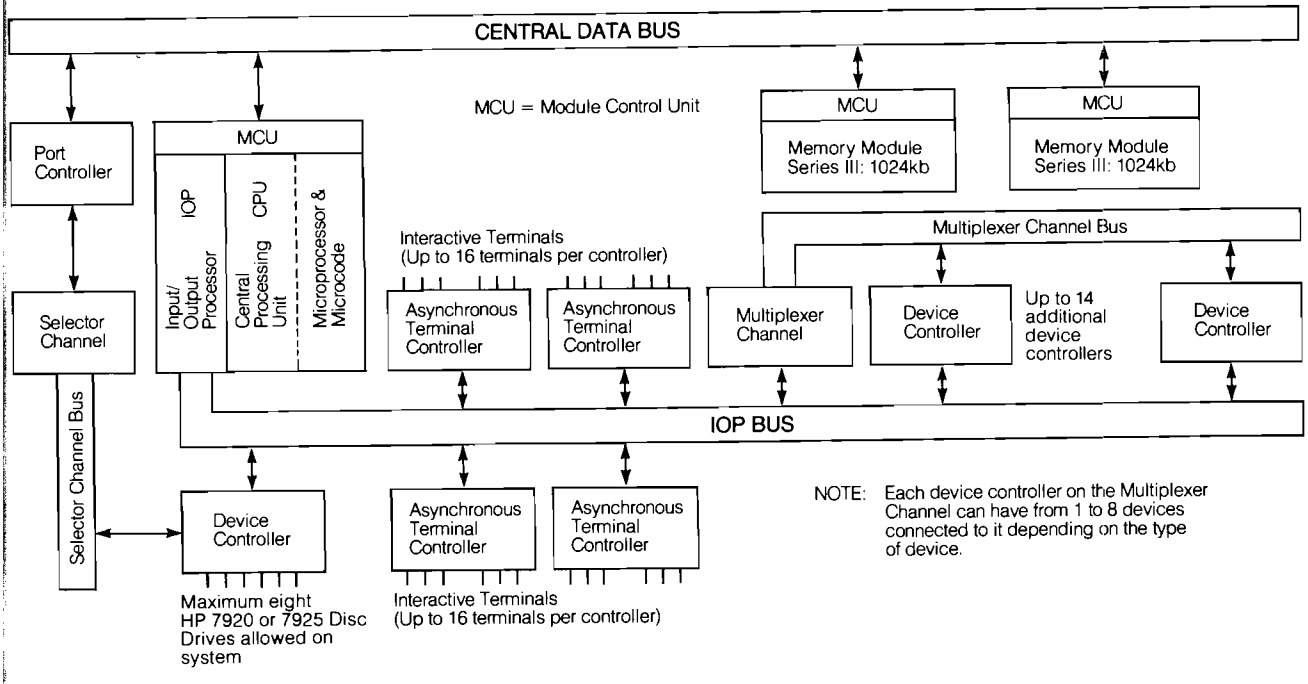
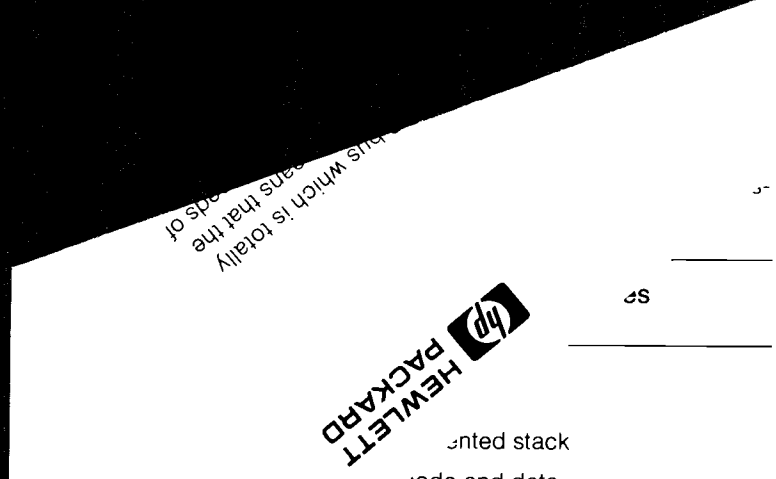


FIGURE F-1. HP 3000 HARDWARE ORGANIZATION (MAXIMUM CONFIGURATION)



- ...nted stack
- ...ode and data
- ...iable, re-entrant code
- ...le-length code segmentation
- Virtual memory for code
- Dynamic relocatability of programs

Implementation

- Microprogrammed CPU
- 175 nanosecond microinstruction time
- Automatic restart after power failure
- Central data bus
- Bus parity checking
- Concurrent CPU and I/O operations

Instructions

- 209 powerful instructions
- All instructions except stack operations are 16 bits in length (stack operations may be packed two per word)
- 16- and 32-bit integer arithmetic
- 32- and 64-bit floating point arithmetic
- 28-digit packed decimal arithmetic
- Special instructions that optimize the efficiency of the operating system.

The three major components of the CPU are the instruction decoder, firmware storage and control, and hardware processor.

The instruction decoder unit converts an instruction in the current instruction register (CIR) into a starting address for the microcode contained in a read-only memory (ROM), and determines various initial conditions required for executing the instruction. As the current instruction is being executed, the next instruction is fetched and placed into the next instruction register (NIR). Upon completion of the current instruction, the contents of NIR are loaded into CIR and the cycle is repeated. This "pipelining" of the current instruction execution with the next instruction-fetch improves throughput by overlapping operations. The HP 3000 III Series instruction set is presented in Appendix G. All instructions are 16 bits in length except stack operations, which are 8-bit instructions. These include a variety of memory reference, branch, arithmetic and data manipulation instructions that operate on integer, real, logical, packed decimal, character and string data. Floating point arithmetic can be performed in single precision (32 bits) or double precision (64 bits), integer arithmetic in 16-bit and 32-bit lengths, and packed decimal instructions extend to 28 digits of precision. In addition, there are a number of instructions designed to aid in creating the multi-programming environment of the system. These include procedure call and exit instructions and others which implement various operating system functions previously done in software.

Firmware storage and control consists of microcode, stored in read-only memory (ROM), and associated control logic. Microcode routines control the operation of the instruction decoder and the hardware processor, in order to create the HP 3000 operating environment—including the 209 instructions available to the programmer. The control storage has a cycle time of 175 nanoseconds and an average microinstruction execution time of 175 nanoseconds.

The hardware processor consists of an arithmetic-logic unit, shifting network, 38 specific purpose registers—20 of which are accessible to user programs, and related data manipulating and testing logic. Since the HP 3000 architecture (see Chapter 6) is structured on code segments and data segments, most of the CPU registers are used for defining the segment limits and operating elements within the segments. As shown in Figure F-2, three of the CPU registers point to locations in a code segment defined as the current code segment. Six of the registers point to locations in a data segment defined as the current data segment. Table F-2 lists all 38 registers and their associated functions. The four top of stack registers are of special interest. In order to improve execution speed, up to four elements from the top of a data stack may be contained in these registers. This allows many functions to be treated as register-to-register operations rather than the slower speed memory-to-register or register-to-memory type operations. These registers are manipulated by the CPU, and their use is fully transparent.

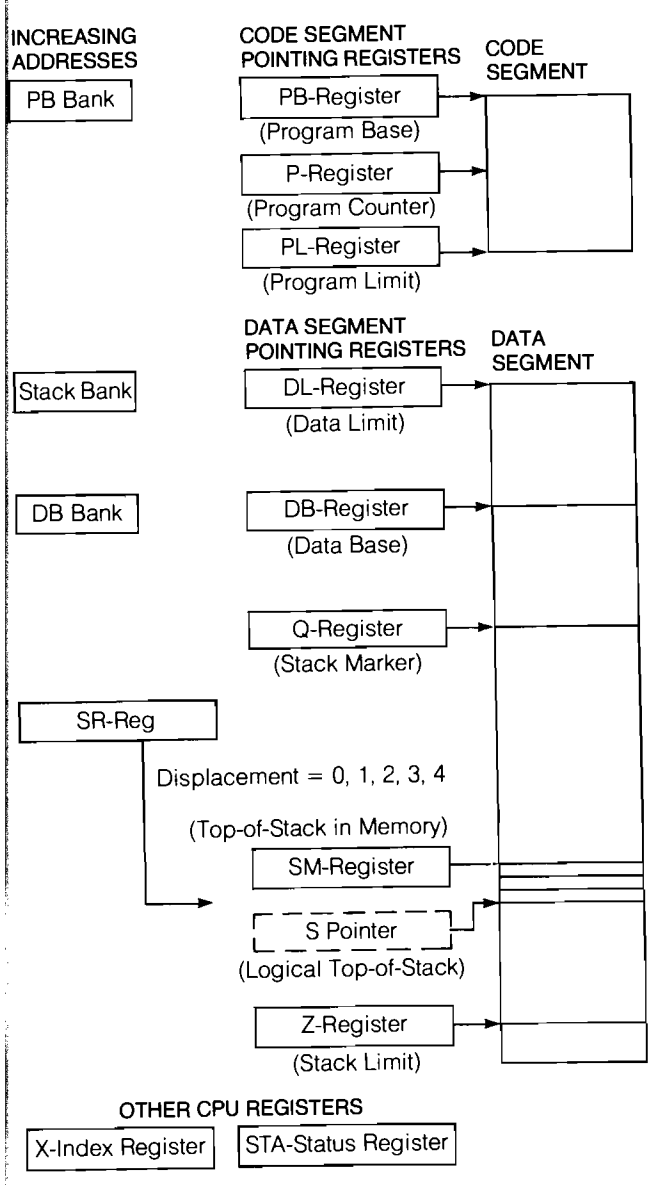


Figure F-2. HP 3000 CPU REGISTERS

TABLE F-2. HP 3000 Hardware Registers

Registers accessible to user programmers

Register	Function
PB, P, PL, PB-Bank	Code Segment Pointers
DL, DB, Q, SM, SR, Z, DB-Bank, S-Bank	Stack Pointers
RA, RB, RC, RD	Top of Stack Registers
X	Index Register
STA	Status Register
SWCH	Switch Register
PCLK	Program Clock Register

Registers dedicated for system use

CIR	Current Instruction Register
NIR	Next Instruction Register
SP0, SP1, SP2, SP3, CTR, ABS-Bank, CPX1, CPX2, MOD	Scratch Pad, Flag, and Interrupt Registers
IOA, IOD	I/O Registers
ACOR, DCOR, OPND	Memory Address and Data Registers
RAR, SAVE	Firmware Address Registers

Main memory

The significant features of the HP 3000 main memory are listed in Table F-3.

TABLE F-3. HP 3000 Main Memory Features

- High-speed, semiconductor, random access memory
- Automatic fault detection and fault correction
- Memory sizes ranging from 256k bytes to 2 megabytes
- Write: 700 nsec minimum cycle time
- Read: 350 nsec access, 700 nsec cycle time
- 45 minute rechargeable battery packs to maintain memory data during power failure

The HP 3000 uses high-speed, semiconductor, random access memory (16k MOS RAM) which provides single-bit fault correction. All double bit faults are detected on the Series III. The parity system is retained with this feature, thus maintaining integrity over the various buses.

Due to the modular design of the HP 3000, any system can be easily expanded from one memory size to another. When there is more than one megabyte of memory in the Series III, it is divided into two modules: one megabyte in the first module and the remainder in the second. These modules can operate concurrently which improves execution time.

The word length transmitted over the bus is 17 bits—16 bits of data (one word or two bytes) and one parity bit. In the memory modules the word length is expanded to 22 bits—16 bits of data and six bits for automatic fault detection and correction.

All detected memory faults are automatically logged to special (non-user) storage. MPE periodically reads this storage and writes the information to the disc file. This file is accessed by an HP Customer Engineer, from a terminal on the system, while performing preventative maintenance. If memory chips have a history of failures, they are replaced during maintenance.

Operating power for the memory modules is supplied by rechargeable battery packs in the semiconductor memory power supplies. When the power supply input voltage is removed, battery power is available for up to 90 minutes (depending on memory size and battery condition) to maintain memory data.

The memory modules interface with other system modules by way of the central data bus. The other system modules may request transfers of data to or from the memory modules on that bus. Operation of the memory modules with other system modules on the central data bus is controlled by module control units, one for each module.

Input/output

All access to input/output devices is by way of the device-independent MPE file system. All location of data, buffering, data transfers, and deblocking are handled automatically by MPE. When you ask to read a named file, you are only implicitly specifying the actual disc address of the file, the file system determines the explicit address and performs the read. At another level, when you ask the file system for a certain type of device by specifying a device class name (e.g., magnetic tape, line printer, etc.), the file system takes care of allocating an actual device. If you must have actual contact with specific devices, you may address them directly. Below this single, flexible interface is a powerful and carefully balanced hardware/software input/output system.

All devices can be operated concurrently (within system bandwidth). Peripherals that fail are taken off-line from the operating system by operator command.

There are two distinct means of implementing I/O: direct I/O (DIO) and programmed I/O (SIO).

Direct I/O (DIO): Direct I/O allows for single word transfers of data, status, or control information between a device connected to the IOP bus and the top of your data stack.

Programmed I/O (SIO): Programmed I/O can be used with devices on either the selector channel (high-speed devices) or the multiplexer channel (medium-to low-speed devices). With programmed I/O, the CPU simply issues a "Start I/O" instruction to the device controller. The device controller, in turn, initiates the SIO program for the particular device which then runs under the control of the selector channel or under the control of the multiplexer channel and the IOP. The SIO program uses a unique set of commands (optimized for I/O operations) to transfer information between main memory and the external device. Both the selector channel and the multiplexer channel (via the IOP) have direct access to main memory. The SIO program and CPU processing run concurrently until the appropriate I/O command terminates the device transfer. This I/O command can also cause an interrupt signal to be sent to the CPU, thus informing the CPU that the I/O task is complete.

Input/output processor (IOP): The input/output processor controls the IOP bus and interrupt lines, providing the communications and data path between the CPU and I/O devices for direct I/O operations and interrupt processing. It also offers a data path between memory and I/O devices for programmed I/O operations.

I/O operations are divided into three categories, direct I/O, programmed I/O, and interrupt processing. Direct I/O operations take place as a result of I/O instructions executed by the CPU; they result in transfer of a word of information between the CPU and an I/O device through the IOP or cause a control function to take place in the I/O system. Devices connected to a multiplexer (see Peripheral I/O Hardware) may use programmed I/O. Once started, programmed I/O operations can, (through the execution of I/O programs stored in memory) transfer block(s) of data between I/O devices and memory, and perform other device control functions without further CPU intervention or attention. The IOP also accepts interrupt requests from the device controllers, interrogates them by means of a poll line to find the highest priority request, and sends an interrupt signal together with the number of the interrupting device to the CPU.

Peripheral I/O hardware: HP 3000 Series III peripheral I/O hardware consists of the selector channel, IOP bus, interrupt lines, multiplexer channel, device controllers, and the peripheral units. Asynchronous terminal controllers interface log-on and data-entry terminals. Table F-4 lists the features which this peripheral I/O hardware offers.

TABLE F-4. HP 3000 Peripheral I/O Hardware Features

- 16 ports per terminal controller
- Up to 4 terminal controllers per system
- Support for 16 device controllers on the multiplexer channel
- Options for type 202S modems
- Type 103A, 202T and 212 type modem support

Asynchronous terminal controller: The asynchronous terminal controller (ATC) is designed to interface terminals to the HP 3000 via the IOP bus. Up to 16 terminals (including the system console) can be interfaced via one ATC, and up to 4 ATCs may be connected to the system. Terminals can be hardwired or connected through type 103A2, 103A3, 103J, 113B, 202S/T, 212 and Vadic VA 3400 modems. Terminals interfaced through the ATC can be configured to the Multiprogramming Executive as data entry terminals under user program control, or as log-on terminals accessing all the capabilities of the system. Since the system console occupies one terminal port, fifteen additional terminal ports are available before expansion to another controller is necessary.

Selector channel: The selector channel interfaces high speed peripheral devices to the HP 3000. Connecting to the central data bus through a port controller, it can accommodate up to eight disc drives connected to one device controller. The selector channel data transfers bypass the IOP completely to provide data transfer rates of up to 2.86 million bytes per second for a single device. Unlike the multiplexer channel (see next section) which switches between device controllers on demand, based on hardware priority, the selector channel maintains the connection for one device controller until it has completed the I/O program. Thus only one I/O program is current at a time for one selector channel. All selector channel data transfers are performed in block mode and the data is applied directly to main memory via the central data bus.

Multiplexer channel: The multiplexer channel acts as a switch to enable one of the 1 through 16 device controllers connected to it to transfer one word of data to or from memory via the IOP, then to allow another controller—based on priority—to perform its transfer. Operating in conjunction with the IOP, the multiplexer allows the device controllers connected to it to run concurrently, interleaving their transfers on a word-by-word basis. By multiplexing, asynchronous cumulative data rates of up to 1,038,000 bytes per second (inbound) and 952,000 bytes per second (outbound) are possible. Data from the multiplexer channel is supplied directly to the IOP for transfer to main memory via the central data bus.

A solid state memory in the multiplexer is divided into 16 sections, one for each device controller. Typically, this memory contains the current I/O program word and related information for each device. When a device is selected for service, the multiplexer executes the indicated operation (or portion thereof) in conjunction with the device controller.

Device controller: The device controller is the hardware linkage between a peripheral device and the computer system. Its primary function is to translate programmed I/O commands from a multiplexer channel or selector channel (or direct I/O commands from the I/O processor) to the unique signals required to control a particular device. When an I/O program is in execution, the device controller responds to and requests service from the multiplexer channel or selector channel. The device controller also generates interrupts when required by some device condition or by direct or programmed command.

Device reference table (DRT): Device controllers are identified by a device number which is used to access the device reference table (DRT). The DRT is known to both hardware and software containing, among other things, a pointer to the start of the SIO program for each device controller. Since there can be a maximum of 125 entries in this table, the HP 3000 logic design allows for up to 125 device controllers in its I/O system (the actual limitation is the 7-bit I/O address bus). Certain device controllers may control several devices. In such cases, each device attached to the controller is addressed separately using a unit number assigned when the device is installed.

Data service and interrupt priorities: In addition to a device number, there are two other characteristic numbers associated with each device. These are the data service priority and interrupt priority. Each of these values is completely independent of the other, and neither is related to the physical location of devices or controllers. This mutual independence of characteristics provides the following advantages:

1. Device numbers can be assigned consecutively, starting with number 3 and proceeding up to the last assigned device in the system. When a new device controller is added, it is merely assigned the next available number (or any vacant number).
2. A new device added to the system may have its controller connected anywhere in the interrupt or data service priority chains, independent of physical location within the cabinet.
3. Since data service priority and interrupt priority are independent of each other, a device which requires a high data transfer rate but interrupts infrequently (such as a disc) may be assigned a high data service priority and a low interrupt priority.

Interrupt system: The interrupt system provides for up to 125 external interrupt levels. When interrupts occur, the microprogrammed interrupt handler identifies each interrupt and grants control to the highest priority interrupt. Current operational status is saved by the microprogram, which then sets up the interrupt processing environment and transfers control to the interrupt routine.

Interrupt routines operate on a common stack (interrupt control stack) which is known to both hardware and software. This feature permits nesting of interrupt routines in the case of multiple interrupts, thus allowing higher priority devices to interrupt lower priority devices.

The interrupt system also provides for 22 internal interrupts (for user errors, system violations, hardware faults, and power fail/restart) plus fourteen traps for arithmetic errors and illegal use of instructions.

Peripherals: Peripheral devices receive output data for storage or display, or supply input data to the computer. Usually, one device controller controls one peripheral device; however, some device controllers are capable of controlling several devices.

Hewlett-Packard furnishes available peripherals as complete I/O subsystems (including the device, interface, cables, etc.) to facilitate system expansion.

For a complete configuration of the supported peripherals on the HP 3000 Series III, refer to the current HP 3000 Price/Configuration Guide.

Automatic restart after power failure

An integral part of the HP 3000 power supply is a power fail/automatic restart capability. When the system AC line voltage falls below 170 volts, the system initiates a power fail warning (PFW). During PFW the system (hardware and MPE) writes all register contents to a reserved section of main memory, activities in the system are successfully completed, and then the power down signal is generated and the system is shut down. The battery back-up power supply refreshes main memory and ensures its validity for up to 45 minutes, depending on memory size and battery condition.

The system is automatically restarted when all power supply voltages reach 90% of their normal values. There is a minimum restart delay of 0.6 seconds following the power on signal before the system is restarted, during which another power failure is possible. After the delay all register values are automatically restored and processing resumes.

Diagnostics

Several levels of diagnostic software help identify and diagnose hardware problems in the HP 3000 computer system. The levels of diagnostics are:

- Verification programs for peripherals run under MPE
- Stand-alone diagnostics to verify all system modules
- Panel microdiagnostics in PROM for CPU, memory, and I/O

Appendix G: HP 3000 Series III Machine Instructions

HEWLETT PACKARD

STACK OP INSTRUCTIONS

ADAX	Add A to X
ADBX	Add B to X
ADD	Add A to B
ADXA	Add X to A
ADXB	Add X to B
AND	Logical AND of A and B
BTST	Test byte on TOS and set CC
CAB	Rotate A-B-C
CMP	Integer compare B, A and set CC
DADD	Double integer add D, C + B, A
DCMP	Double integer compare and set CC
DDEL	Double delete TOS
DDIV	Double integer divide
DDUP	Double duplicate TOS
DECA	Decrement A
DECB	Decrement B
DECX	Decrement X
DEL	Delete TOS
DELB	Delete B
DFLT	Float a double integer
DIV	Integer divide B by A
DIVL	Divide long integer C, B ÷ A
DMUL	Double integer multiply
DNEG	Double integer negate
DSUB	Double integer subtract D, C − B, A
DTST	Test double word on TOS and set CC
DUP	Duplicate TOS
DXCH	Double exchange
DZRO	Push double zero onto stack
FADD	Floating point add D, C + B, A
FCMP	Floating point compare and set CC
FDIV	Floating point divide D, C ÷ B, A
FIXR	Fix and round
FIXT	Fix and truncate
FLT	Float an integer
FMPY	Floating point multiply
FNEG	Floating point negate
FSUB	Floating point subtract D, C—B, A
INCA	Increment A
INCB	Increment B
INCX	Increment X
LADD	Logical add A + B
LCMP	Logical compare B, A and set CC
LDIV	Logical divide C, B ÷ A
LDXA	Load X into A
LDXB	Load X into B
LMPY	Logical multiply B × A
LSUB	Logical subtract B − A
MPY	Multiply integers, integer product
MPYL	Multiply integers, long integer product
NEG	Integer negate
NOP	No operation
NOT	Logical complement TOS
OR	Logical OR of A, B
STAX	Store A into X
STBX	Store B into X
SUB	Integer subtract B − A
TEST	Test TOS and set CC
XAX	Exchange A and X
XBX	Exchange B and X
XCH	Exchange A and B
XOR	Logical exclusive OR of A, B
ZERO	Push integer zero onto stack
ZROB	Zero B
ZROX	Zero X

SHIFT INSTRUCTIONS

ASL	Arithmetic shift left
ASR	Arithmetic shift right
CSL	Circular shift left
CSR	Circular shift right
DASL	Double arithmetic shift left
DASR	Double arithmetic shift right
DCSL	Double circular shift left
DCSR	Double circular shift right
DLSL	Double logical shift left
DLSR	Double logical shift right
LSL	Logical shift left
LSR	Logical shift right
QASL	Quadruple arithmetic shift left
QASR	Quadruple arithmetic shift right
TASL	Triple arithmetic shift left
TASR	Triple arithmetic shift right
TNSL	Triple normalizing shift left

LEGEND

TOS	Top of stack	A	Top of stack	D	Location below C
CC	Condition Code	B	Location below A	DB	Data Base
X	Index Register	C	Location below B	DL	Data Limit

FIELD AND BIT INSTRUCTIONS

DPF	Deposit field, A bits to B	TCBC	Test and complement bit, set CC
EXF	Extract specified field, right-justify	TRBC	Test and reset bit, set CC
SCAN	Scan bits	TSBC	Test and set bit, set CC
TBC	Test specified bit and set CC		

BRANCH INSTRUCTIONS

BCC	Branch on specified CC	BRO	Branch on TOS odd (bit 15 = 1)
BCY	Branch on carry	CPRB	Compare range and branch
BNCY	Branch on no carry	DABZ	Decrement A, branch if zero
BNOV	Branch on no overflow	DXBZ	Decrement X, branch if zero
BOV	Branch on overflow	IABZ	Increment A, branch if zero
BR	Branch unconditionally	IXBZ	Increment X, branch if zero
BRE	Branch on TOS even (bit 15 = 0)		

MOVE INSTRUCTIONS

CMPB	Compare bytes in two memory blocks	MVB	Move bytes in memory, addresses +/−
MABS	Move using absolute addresses		
MDS	Move using data segments	MVBL	Move words from DB+ to DL+ area
MFDS	Move from data segment	MVBW	Move bytes while of specified type
MOVE	Move words in memory, addresses +/−	MVLB	Move words from DL+ to DB+ area
		SCU	Scan bytes until test or terminal byte
MTDS	Move to data segment	SCW	Scan bytes while equal to test byte

PRIVILEGED MEMORY REFERENCE INSTRUCTIONS

LDEA	Load double word from extended address	PSTA	Privileged store into absolute address
		SDEA	Store double word into extended address
LSEA	Load single word from extended address		
		SSEA	Store single word into extended address
LST	Load from system table		
PLDA	Privileged load from absolute address	SST	Store into system table

IMMEDIATE INSTRUCTIONS

ADDI	Add immediate to integer in A	LDXI	Load X immediate
ADXI	Add immediate to X	LDXN	Load X negative immediate
ANDI	Logical AND immediate with A	MPYI	Multiply immediate with A
CMPI	Compare A with immediate, set CC	ORI	Logical OR immediate with A
CMPN	Compare A with negative immediate	SBXI	Subtract immediate from X
DIVI	Divide immediate into A	SUBI	Subtract immediate from A
LDI	Load immediate to TOS	XORI	Logical exclusive OR immediate
LDNI	Load negative immediate to TOS		

REGISTER CONTROL INSTRUCTIONS

ADDS	Add operand to stack pointer	SETR	Set specified registers from stack
PSHR	Push specified registers onto stack	SUBS	Subtract operand from stack pointer
RCLK	Read clock	XCHD	Exchange DB and TOS
SCLK	Store clock		

PROGRAM CONTROL AND SPECIAL INSTRUCTIONS

DISP	Dispatch
EXIT	Exit from procedure
HALT	Halt
IXIT	Interrupt exit
LLBL	Load label
LLSH	Linked list search
LOCK	Lock resource
PAUS	Pause, interruptable
PCAL	Procedure call
PCN	Push CPU code number
PSDB	Pseudo interrupt disable
PSEB	Pseudo interrupt enable
RSW	Read switch register
SCAL	Subroutine call
SXIT	Exit from subroutine
UNLK	Unlock resource
XEQ	Execute stack word

I/O INSTRUCTIONS

CIO	Control I/O, direct
CMD	Send command to module, direct
RIO	Read I/O, direct
RMSK	Read device mask
SED	Set enable/disable external interrupts
SIN	Set interrupt
SIO	Start I/O, block transfer
SMSK	Set device mask
TIO	Test I/O, direct
WIO	Write I/O, direct

LOOP CONTROL INSTRUCTIONS

MTBA	Modify variable test against limit, branch
MTBX	Modify X, test against limit, branch
TBA	Test variable against limit, branch
TBX	Test X against limit, branch

MEMORY ADDRESS INSTRUCTIONS

ADDM	Add memory to TOS
CMPM	Compare TOS with memory
DECM	Decrement memory
INCM	Increment memory
LDB	Load byte onto stack
LDD	Load double word onto stack
LDPN	Load double from program, negative
LDPP	Load double from program, positive
LDX	Load X
LOAD	Load word onto stack
LRA	Load relative address onto stack
MPYM	Multiply TOS by memory
STB	Store byte on TOS into memory
STD	Store double on TOS into memory
STOR	Store TOS into memory
SUBM	Subtract memory from TOS

EXTENDED INSTRUCTION SET

Extended-Precision Floating Point

EADD	Add
ECMP	Compare
EDIV	Divide
EMPY	Multiply
ENEG	Negate
ESUB	Subtract

Decimal Arithmetic

ADDD	Decimal add
CMPD	Decimal compare
CVAD	ASCII to decimal conversion
CVBD	Binary to decimal conversion
CVDA	Decimal to ASCII conversion
CVDB	Decimal to binary conversion
DMPY	Double logical multiply
MPYD	Decimal multiply
NSLD	Decimal normalizing left shift
SLD	Decimal left shift
SRD	Decimal right shift
SUBD	Decimal subtract

Appendix H: HP 3000 Guide to Synchronous Modems

HEWLETT PACKARD

Selection of a modem is a critical factor in planning a geographically dispersed network. A wide variety of modems is available, so this guide has been prepared to aid in choosing the proper unit. Hewlett-Packard recommends that the modems listed below be used. If other modems are selected, they must be functionally and electrically identical (at the system interface) with the recommended modems.

Modem selection criteria

HP synchronous modems are fully compatible with the synchronous communication subsystems available on the HP 3000 computers. They are not available in all countries, and your local sales office should be consulted. The selection, installation, and proper functioning of common carrier or third party modems is your responsibility. Hewlett-Packard accepts responsibility for maintaining hardware and software compatibility only with recommended modems. To determine the optimum modem for your system, the following parameters must be considered.

Operating mode: Switched network (dial-up) or leased (private) line.

Type of circuit: Two-wire (half duplex) or four wire (full duplex).

Transmission speed: 1200, 2400, 4800, or 9600 bps.

Line conditioning: Unconditioned or conditioned.

A communications network's performance, in terms of throughput and responsiveness, is heavily influenced by the decisions made concerning these parameters. For low volume networks, low speed communications may be appropriate. Most users normally will select the highest performance service that meets their budgetary requirements. Since modem selection is also based on software related considerations, you should consult your Hewlett-Packard representative regarding the HP communications software and applications which you will be using.

Operating mode: Switched network (dial-up) operation normally is advisable when communications occur only periodically (once or twice per day) and the volume of data to be transmitted is low to moderate. Generally, only two-wire circuits are available for switched networks, with operation limited to 4800 bps or below.

Leased line service is appropriate for higher volumes of data and continuous on-line service.

Type of circuit: Two-wire communications circuits send/receive in one direction at a time, and to reverse the direction of transmission, a "line turnaround" must be performed. Since frequent, time-consuming turnarounds are necessary, two-wire circuits should be selected only for low volume/response applications.

Four-wire circuits can send and receive simultaneously, which eliminates turnaround delays. Most leased line service is four-wire. To maintain responsiveness and the best transmission throughput, four-wire circuits are suggested.

Transmission speed: Transmission speed is the most critical decision to be made in selecting communications facilities for distributed systems networks. Usually the speed of the service, including circuits and modem, is directly associated with the cost; higher speed means higher cost. Availability of service from the common carrier, the volume of traffic, and the need for responsiveness during interactive access are key factors in selecting the transmission speed.

In many areas, only leased-line service is available to 9600 bps on voice-grade circuits.

- In most countries 4800 bps leased-line service is offered.
- Some countries also allow 4800 bps service on a switched circuit (dial-up) basis, although usually with only two-wire circuits.
- Nearly all countries provide both switched and non-switched (leased line) service at 2400 bps.

Line conditioning: By applying certain internal compensation to a communications circuit, the common carrier can remove noise and other degrading characteristics to improve the quality of the circuit. The level of conditioning recommended for the modem should always be used to ensure reliable service.

Recommended modems

Hewlett-Packard modems operate with DS/3000, RJE/3000, MRJE/3000, and MTS/3000. The modems listed below may be used:

FOR CANADA AND THE UNITED STATES

Switched service (dial up)

HP 37210T	(4800 bps)
Bell 201C	(2400 bps)
Bell 208B	(4800 bps)

Leased line service

HP 37210T	(4800 bps)
HP 37220T	(9600 bps)
Bell 201C	(2400 bps)
Bell 208A	(4800 bps)
Bell 209A	(9600 bps)

If you choose other modems you are responsible for working with their supplier to ensure that the units are fully equivalent to the HP recommended equipment.

For international areas

In many countries, the provision of telecommunications services is controlled by a local transmission authority. Check with your local Hewlett-Packard representative regarding the availability of Hewlett-Packard modems. With other modems, the user or Hewlett-Packard must demonstrate satisfactory operation with the locally supplied modem prior to HP's acceptance of responsibility for compatible operation with the supplied communications software.

Index

For more information on HP 3000 Computer Systems, contact your local Hewlett-Packard representative, or write

Hewlett-Packard
General Systems Division
Marketing Department
19447 Pruneridge Avenue
Cupertino, CA 95014
Telephone (408) 725-8111

In Europe: Hewlett-Packard S.A.
7, rue du Bois-du-Lan,
P.O. Box CH-1217 Meyrin 2
Geneva, Switzerland
Telephone: (022) 82 70 00

In Japan: Yokogawa-Hewlett-Packard Ltd.
29-21 Takaido Higashi 3-chome
Suginami-ku, Tokyo 168
Telephone (03) 331-6111

In Canada: Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
Mississauga, Ontario L4V 1M8
Telephone (416) 678-9430

Other International Locations:
Hewlett-Packard
3495 Deer Creek Road
Palo Alto, CA 94304 U.S.A.
Telephone (415) 856-1501

HEWLETT PACKARD

19447 Pruneridge Avenue, Cupertino, California 95014

Part No. 30000-90008
5953-0589
Printed in U.S.A. 6/80