HP 3000 Series II Computer System

General Information Manual

Computer Museum

HEWLETT hp PACKARD

5303 STEVENS CREEK BLVD., SANTA CLARA, CALIFORNIA, 95050

Part No. 30000-90008

Printed in U.S.A. 6/77

This 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.

This document contains proprietary information which is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced or translated to another program language without the prior written consent of Hewlett-Packard Company.

Copyright © 1977 by HEWLETT-PACKARD COMPANY

SECOND EDITION

HP Computer Museum
www.hpmuseum.net

For research and education purposes only.

PREFACE

There are twelve manuals that together document the Multiprogramming Executive II operating system (MPE-II) for the HP 3000 Series II Computer System. The diagram on the following page, which is repeated in the Preface of each operating system manual, shows the relationship of the manual you are holding (shaded block) to others in the set. As indicated by the diagram, the twelve manuals in the set are organized by function into four levels:

- INTRODUCTORY LEVEL manuals

  These two manuals introduce all the major concepts of the HP 3000 Series II Computer System and provide an introduction to the use of the system.

- STANDARD USER LEVEL manuals

  These seven manuals provide the detailed reference documentation for the entire operating system — excluding only the system administration functions. Of these manuals, the *Commands Reference Manual* is essential for using the HP 3000 Series II. The *Error Messages and Recovery Manual* summarizes the error messages and recovery actions for operating system and language/utility subsystem errors. The *Index to MPE Reference Documents* is a master index for the entire set of MPE-II operating system manuals. The other four manuals cover more advanced programmatic features, going still more deeply into the operating system itself.

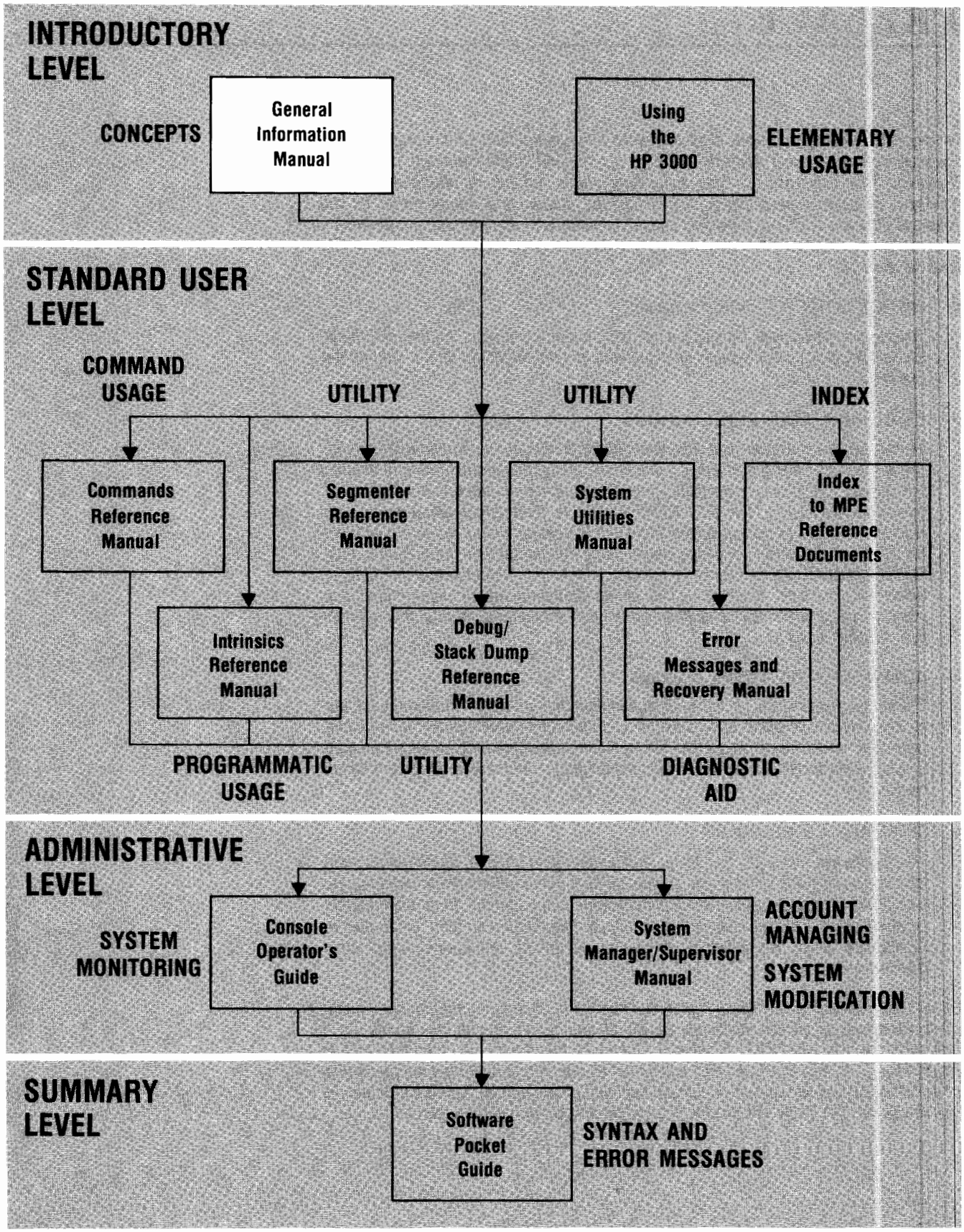
- ADMINISTRATIVE LEVEL manuals

  These two 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 — such as setting up an accounting system and modifying the operating system when necessary.

- SUMMARY LEVEL manual

  The *Software Pocket Guide* summarizes the MPE-II commands and intrinsics (system procedures), the MPE-II File System error messages, and the language and utility subsystem commands, in a convenient pocket size. This guide is primarily a memory aid and assumes a familiarity with the effects of the commands, intrinsics, and their parameters.

All of the available operating system, machine reference, language, utility, data base management, data communications, and educational application manuals for the HP 3000 Series II Computer System are summarized in Appendix C of this manual. Included in that summary are the complete title of each manual, the Hewlett-Packard manual part number (used for ordering manuals), and a brief description of what each manual is about.


INTRODUCTORY LEVEL
CONCEPTS
General Information Manual
Using the HP 3000
ELEMENTARY USAGE
STANDARD USER LEVEL
COMMAND USAGE
UTILITY
UTILITY
INDEX
Commands Reference Manual
Segmenter Reference Manual
System Utilities Manual
Index to MPE Reference Documents
Intrinsics Reference Manual
Debug/ Stack Dump Reference Manual
Error Messages and Recovery Manual
PROGRAMMATIC USAGE
UTILITY
DIAGNOSTIC AID
ADMINISTRATIVE LEVEL
SYSTEM MONITORING
Console Operator's Guide
System Manager/Supervisor Manual
ACCOUNT MANAGING
SYSTEM MODIFICATION
SUMMARY LEVEL
Software Pocket Guide
SYNTAX AND ERROR MESSAGES

CONTENTS

SECTION 1 — GENERAL OVERVIEW

System Features 1
Hardware 3
Software 3
System Configurations 4

SECTION 2 — HARDWARE FEATURES

Central Processing Unit (CPU) 6
  Separate Code and Data 7
  Data Stacks 7
  Variable-Length Segmentation 9
  Virtual Memory 9
  Registers 9
  Instructions 10
Microcode 11
Main Memory 11
Input/Output 12
  Direct I/O (DIO) 12
  Programmed I/O (SIO) 12
    Multiplexer Channel 12
    Selector Channel 13
  Device Reference Table (DRT) 13
  Data Service and Interrupt Priorities 13
  Interrupt System 13
Peripherals 14

SECTION 3 — THE OPERATING SYSTEM

Multiprogramming 15
Interactive Processing 16
Batch Processing 17
Spooling 17
MPE Commands 18
MPE Intrinsics 19
The System Administrators 20
Accounting Facility 21
  Coordinating System Access 21
  Coordinating File Usage 21
The File System 22
Processes 24
Scheduling 24
The Segmenter 25
Capability Sets 26
The Logging Facility 26
Start-Up and Modification of MPE 27
System Back-Up and Recovery 27
Power Fail/Auto Restart 28

SECTION 4 — PROGRAMMING LANGUAGES

Multilingual Programming 29
COBOL/3000 30
RPG/3000 31
FORTRAN/3000 33
BASIC/3000 34

Computer Museum

SPL/3000 36
"High Level" SPL Features 36
Machine-Dependent SPL Features 36
Sample SPL Procedure 37
Documentation 38
APL 39
APL Features 39
APL Character Set 39
APL Keyboard 41
Calculator Mode vs Function Mode 41

SECTION 5 — UTILITIES

Text Editor 43
File Copier 46
Sort/Merge 47
Compiler Library 47
Scientific Library 48
Trace Facility 48
Data Entry Library 49

SECTION 6 — DATA MANAGEMENT

KSAM 51
KSAM File Structure 52
KSAM File Access 52
KSAM Components 53
KSAM Features 53
IMAGE 54
IMAGE Data Base Structure 54
Types of Data Sets 58
Using IMAGE 60
Accessing the Data Base 61
Maintaining the Data Base 63
QUERY 64
Applications 65
Features 65
Command Set 66
Using QUERY 67

SECTION 7 — DATA COMMUNICATIONS

Distributed Systems/3000 71
DS/3000 Capabilities 71
Remote Command Processing 71
Remote File Access 72
Program-to-Program Communications (PTOP) 72
2780/3780 Emulator 77
IBM 2780 Data Transmission Terminal 78
IBM 3780 Data Communication Terminal 79
HP 2780/3780 Emulator Capabilities 79
System Environment 80
Documentation 80
Example 80
Real-Time Programmable Controller 82
Features of RTE-C 83
Hardware Components 83
Software Components 84
Applications Software 85
Documentation 85

Programmable Controller 86
Hardware Components 86
Software Components 87
Applications Software 87
Documentation 88

APPENDIX A — MPE-II Commands

APPENDIX B — MPE-II Intrinsics

APPENDIX C — HP 3000 Series II Documentation

APPENDIX D — HP 3000 Series II Machine Instructions

Index

GENERAL OVERVIEW 1

The HP 3000 Series II Computer System is a general purpose computer with true multiprogramming and multilingual capabilities. It can simultaneously handle many interactive and batch operations, each in any of several programming languages. The HP 3000 Series II incorporates such advanced features as a hardware stack architecture, variable-length code segmentation in a hardware-assisted virtual memory scheme, user protection, and dynamic storage allocation.

Computer Museum

The HP 3000 Series II is the result of an integrated hardware-software design effort based on the specific demands of the multiprogramming environment. Hardware and software work together in an interrelated manner with hardware performing many of the operations conventionally performed by software, such as environment changes on interrupts or subroutine calls.

A powerful operating system, the Multiprogramming Executive II (MPE-II), optimizes the concurrent operation of many related or unrelated programs. Such system resources as main memory, the central processor, and peripheral channels and devices are dynamically allocated to each program as needed. Typically each user is independent of all others using the system. For those cases where users need to interact and cooperate on joint efforts, however, the operating system provides extensive facilities for sharing information.

SYSTEM FEATURES

The HP 3000 Series II incorporates many features otherwise found only on very large computer systems. These features are summarized in the next few paragraphs and are discussed more thoroughly in subsequent chapters.

Stack Architecture. Many powerful operating features stem from the use of "data stacks". The stack architecture of the HP 3000 Series II provides private, hardware-protected data storage for each user as well as an automatic method for moving this data to and from the central processor registers. The major benefits of this approach are:

- Fast execution
- Code compression
- Hardware-protected execution
- Dynamic allocation of subprogram data space
- Ease of parameter passing
- Highly efficient subprogram linkage
- Rapid interruption and restoration of user environments
- Recursion (a subprogram being able to call itself)

Microprogrammed Operations. Additional economy has been provided by microprogramming many system operations that in the past were programmed in software. These operations are requested by machine instructions that each, in turn, execute multiple microinstructions built into the central processor hardware. Microprogramming eliminates repetitive cod-

ing otherwise needed for recurring operations such as saving registers on subprogram calls, manipulating characters, and transferring data to and from buffers.

Data Base Management Facilities. The HP 3000 Series II provides powerful software facilities which allow the user to create, access, and maintain large data bases. The information in these data bases can be accessed both interactively from a keyboard terminal and programmatically from user programs written in any of the available programming languages.

Five Programming Languages. The HP 3000 Series II provides the user with a true multilingual programming environment. The languages available are COBOL, RPG, FORTRAN, BASIC, and the HP 3000 Systems Programming Language (SPL). SPL is a language developed especially for the HP 3000 Series II which combines the best features of "high level" and assembly languages. BASIC programs can be developed using either the BASIC Interpreter or the BASIC Compiler.

Virtual Memory. The operating system's hardware-assisted virtual memory scheme offers each user program a very large memory space that far exceeds the maximum main memory size of 524,288 bytes. Virtual memory consists of both main memory and an extensive, flexible storage area on disc. In the HP 3000 Series II, virtual memory has been implemented using a segment trap frequency algorithm which ensures the automatic presence in main memory of only those segments of code and data which are currently required by executing programs. Main memory is thus shared efficiently by many users in a fashion which gives each programmer the impression he is working with a much larger computer system.

Fault Control Memory. The HP 3000 Series II employs high-speed semiconductor memory modules which provide automatic fault detection and single-bit correction with no loss in performance.

Concurrent I/O and CPU Operations. Within the HP 3000 Series II, many input/output operations can be performed concurrently with CPU and memory operations. This is possible because, in addition to the Central Processing Unit (CPU), the HP 3000 Series II has an Input/Output Processor (IOP) with its own dedicated data transfer path (the IOP Bus) to which are connected a Multiplexer Channel and one or more Asynchronous Terminal Controllers. All of this hardware operates under the control of the Multiprogramming Executive II operating system which handles all queuing and device scheduling.

Re-Entrant Code and Private Data. Within the MPE environment, many user and system functions can be active concurrently without interferring with each other. This is because the hardware provides protection of programs and guarantees the privacy of user data areas. The hardware keeps code and data physically separate by organizing them into re-entrant code segments (which can be shared among users but not altered) and data segments (which cannot be shared but which can be altered by the creating user). This segmenting ability greatly facilitates the operation of virtual memory in that 1) code segments need never be swapped out, since an identical copy always exists on disc, and 2) code segments can be swapped indirectly from wherever the program file resides on disc without having to be copied first to a special swapping disc.

A Powerful Operating System. The HP 3000 Series II has a single, comprehensive operating system which supervises the processing of all user programs and which provides the user with an extensive and powerful set of system functions. The major features of the operating system are:

- Fully interchangeable batch and interactive processing
- A uniform, device-independent, and language-independent file system
- File coordination and security
- Spooling of input and output
- Console job control
- User coordination and accounting
- Resource control
- Automatic scheduling (under the control of the installation's management)
- System back-up facility
- Power fail/auto restart
- System tailoring (under the control of the installation's management)
- System logging facility.

HARDWARE

The hardware design of the HP 3000 Series II is discussed in section II of this manual. Briefly, the system offers the following hardware features:

- Up to 512K bytes of high-speed, fault correcting semiconductor memory
- A high-speed selector channel for block transfers between main memory and disc storage
- I/O multiplexer channels for word transfers between main memory and low- to medium-speed devices such as card readers, line printers, and magnetic tape units
- Asynchronous terminal controllers for data transmission between main memory and interactive terminals
- High-speed disc storage devices
- 800 or 1600 byte-per-inch magnetic tape units
- 200, 300, 600, or 1250 line-per-minute line printers
- Cathode ray tube (CRT) display terminals
- Hard copy terminals with speeds up to 120 characters-per second

SOFTWARE

The HP 3000 Series II offers a wide range of software including a powerful operating system, five programming languages, a versatile text editor, and extensive and flexible file copier, a fast and powerful sort/merge package, two libraries of commonly used mathematical, statistical, and utility procedures, a system trace facility, a comprehensive data base management facility, and data communications products. Figure 1-1 shows a block diagram representation of the currently available software. The software shown in that figure is discussed in sections III through VII of this manual.

SYSTEM CONFIGURATIONS

The HP 3000 Series II is available in models which range from a small computer for business to a general purpose data processing computer. User installations can easily increase the overall performance of their system by adding more peripheral devices and by adding more main memory. The same operating system, language processors, utility programs, data base management programs, and data communications programs are used on all models of the HP 3000 Series II. Consequently the transition from one model to another, while it improves system performance, requires no changes to the user's programs.

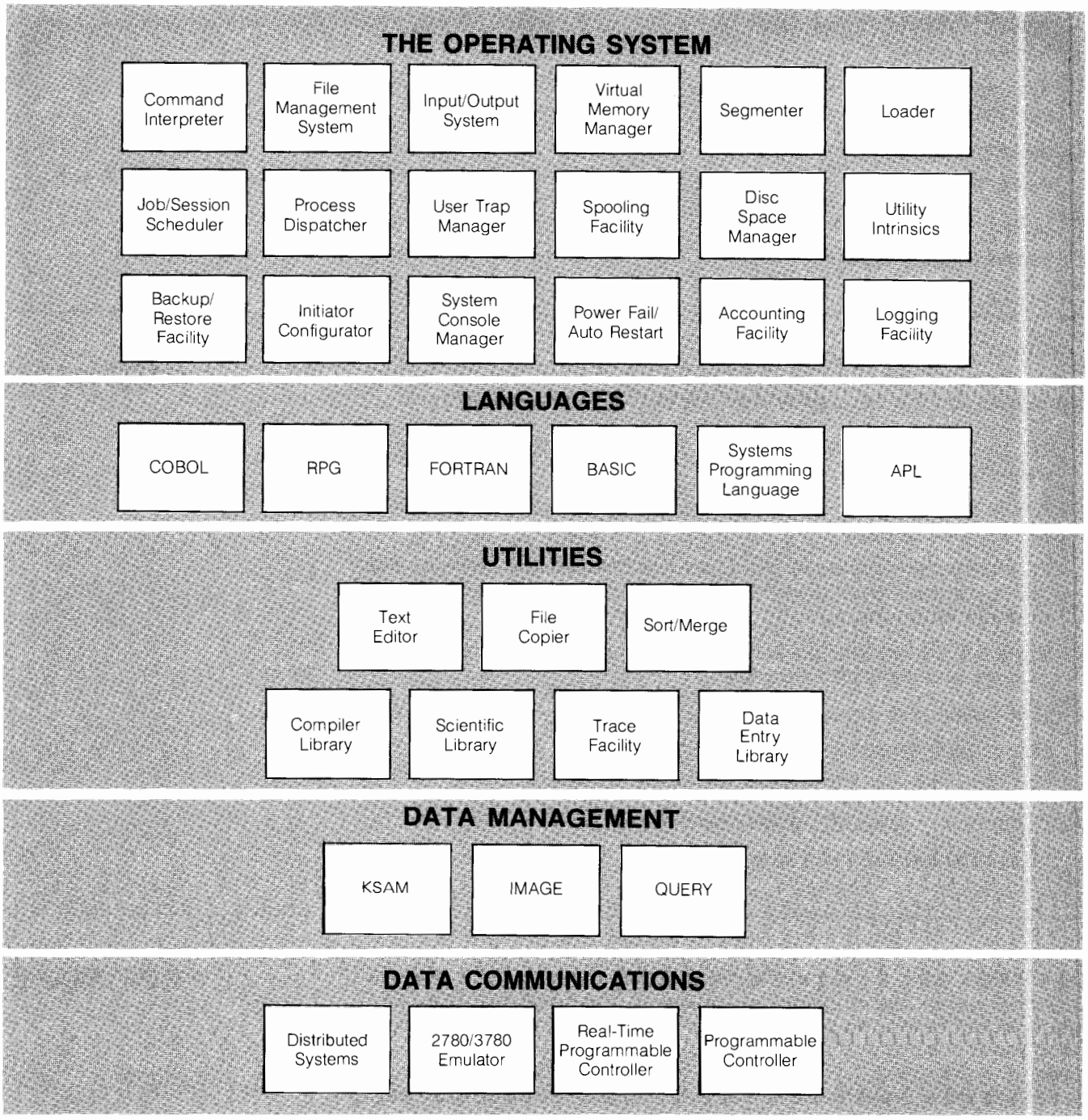


Figure 1-1. HP 3000 Series II Software

HARDWARE FEATURES 2

NOTE

Programmers who deal exclusively with high-level languages such as COBOL and RPG are truly machine-independent in the sense that they need not know the machine instruction set nor the design of the hardware. Programmers who write some or all of their programs in SPL (the Systems Programming Language for the HP 3000 Series II) should, however, have a working knowledge of both the machine instruction set and the architecture of the machine. This section covers both topics in a general manner. The architecture of the machine is described in detail in the *HP 3000 Series II System Reference Manual* (HP part number 30000-90020) and the machine instruction set is described in detail in the *HP 3000 Series II Machine Instruction Set Reference Manual* (HP part number 30000-90022).

This section summarizes the hardware features of the HP 3000 Series II. The design of the hardware provides an efficient and powerful foundation upon which the software is built.

The hardware elements of an HP 3000 Series II are organized as illustrated in figure 2-1. 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

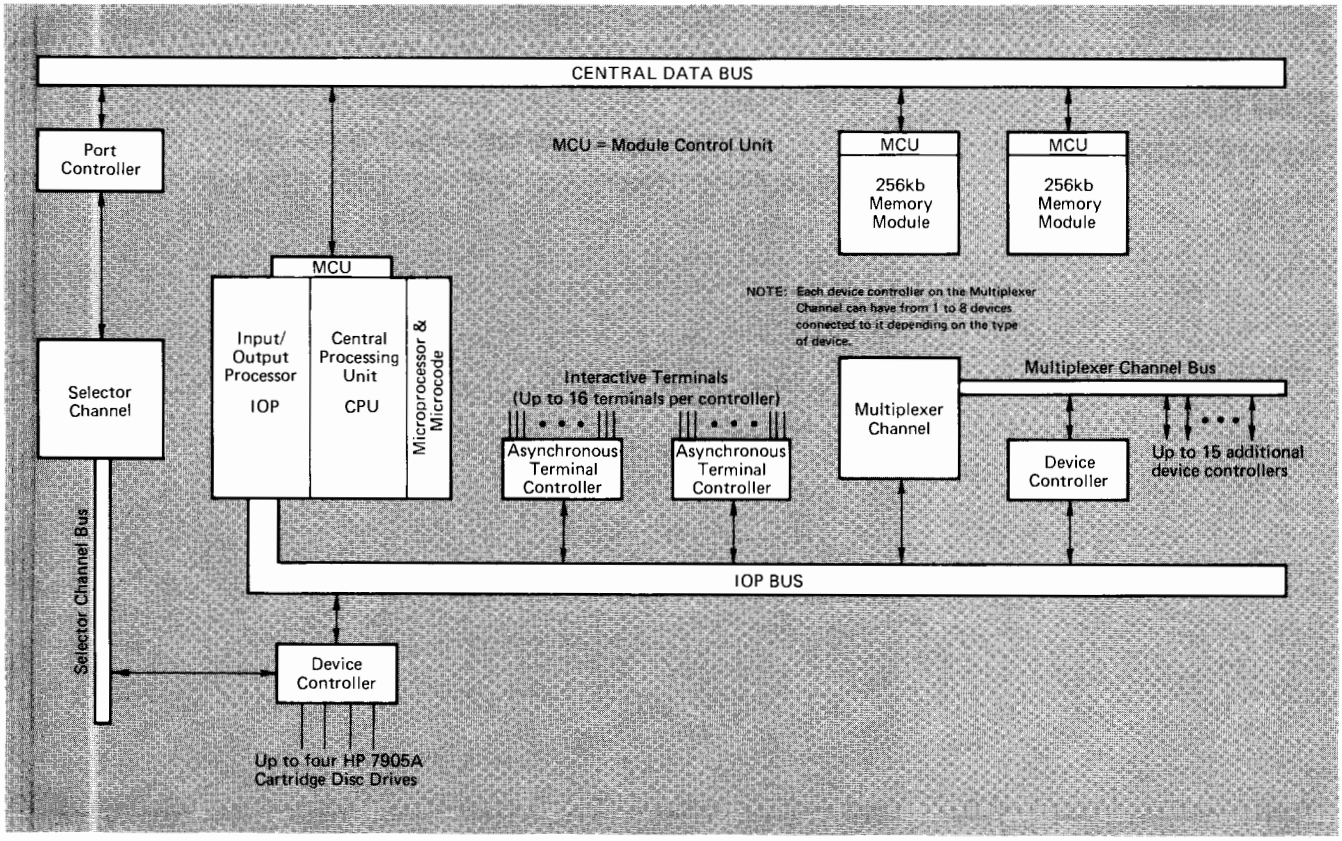


Figure 2-1. HP 3000 Series II Hardware Organization

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 Series II 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. New equipment can also be added without having to go through a major hardware reconfiguration.

The presence of a separate IOP Bus which is totally dedicated to input/output data transfers means that the HP 3000 Series II can always respond immediately to the needs of 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, or, for lower speed devices, it 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.

CENTRAL PROCESSING UNIT (CPU)

The significant features of the HP 3000 Series II CPU are listed in table 2-1.

Table 2-1. HP 3000 Series II CPU Features

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

Separate Code and Data

Code consists of the executable instructions that make up a program or subprogram. The values and arrays used by a program or subprogram are referred to as *data*. In the HP 3000 Series II, 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 non-modifiable while active in the system, permits code to be sharable and re-entrant. *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 same 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 (discussed in the next topic below) 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.

Data Stacks

The data for each user is organized into a *data stack*. In general, a stack is a storage area where the last item stored in is usually the first item taken out. 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 Series II, 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, the stack is manipulated automatically for the user by the language processor. The user can, however, manipulate the stack directly by writing subprograms in SPL.

Figure 2-2 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. 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 further 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

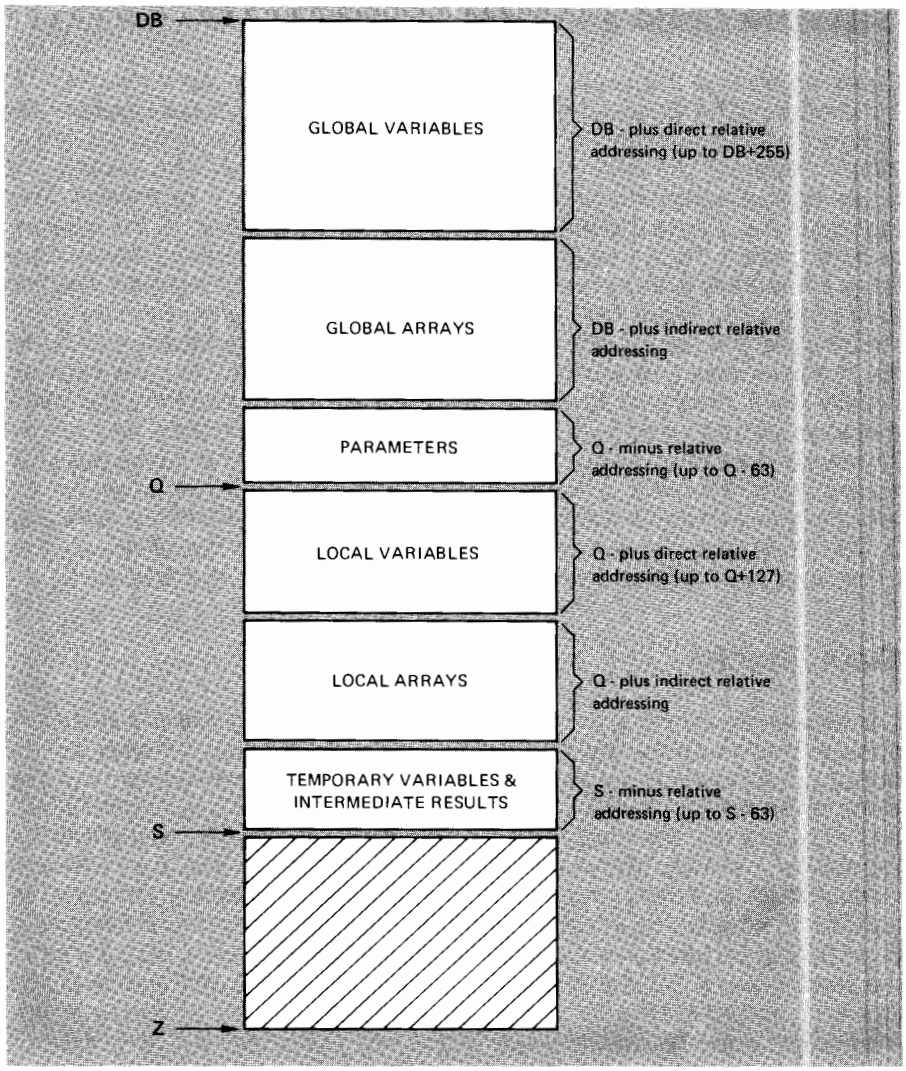


Figure 2-2. Subprogram's View Of The Data Stack

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 four elements of the stack are the most frequently used, there are four CPU registers (RA, RB, RC, and RD) which may contain up to four of 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 stack locations S, S– 1, S– 2, or S– 3 are specifically referenced. Data stacks are expanded automatically by the operating system during execution up to a maximum size of 64K bytes.

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 is needed while the subprogram is executing. The location and size of all executing code segments is maintained in a Code Segment Table while the location and size of all associated data segments is maintained 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 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.

Virtual Memory

The HP 3000 Series II employs a concept called *virtual memory* which greatly expands the amount of memory space available to user's programs. 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 executing, only those of its 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 Series II code segments are non-modifiable and re-entrant; when the segment is again needed, it is simply copied again from the disc on which the program resides. Since programs are copied into main memory directly from the disc on which they reside, 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 data 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 transfering segments between secondary memory and main memory is referred to as *swapping*. In this way many users may have very large programs executing concurrently with one another and with the operating system. The operating system itself and the various language and utility programs are handled in exactly the same manner. On the HP 3000 Series II, the virtual memory scheme uses a working set table to implement an advanced "segment trap frequency" replacement algorithm designed especially for the HP 3000 Series II.

Registers

The architecture of the HP 3000 Series II employs a set of 38 *specific purpose* registers rather than a smaller set of *general purpose* registers. Each register was 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.

Of the 38 registers, 20 are accessible to user's programs and/or the operating system and its related software. The other 18 registers are used, for example, by the interrupt system and for microprogram processing. The registers are summarized in table 2-2.

Table 2-2. HP 3000 Series II Machine Registers

REGISTERS ACCESSIBLE TO THE PROGRAMMER		REGISTERS DEDICATED FOR SYSTEM USE	
Register	Function	Register	Function
PB, P, PL, PB-Bank	Code Segment Pointers	CIR	Current Instruction Register
		NIR	Next Instruction Register
DL, DB, Q, SM, SR, Z, DB-Bank, S-Bank	Stack Pointers	SP0, SP1, SP2, SP3, CTR, ABS-Bank, CPX1, CPX2, MOD	Scratch Pad, Flag, and Interrupt Registers
RA, RB, RC, RD	Top of Stack Registers	IOA, IOD	I/O Registers
		ACOR, DCOR, OPND	Memory Address and Data Registers
X	Index Register		
STA	Status Register	RAR, SAVE	Firmware Address Registers
SWCH	Switch Register		
PCLK	Program Clock Register		

Instructions

There are 209 unique instructions in the HP 3000 Series II. Many of these instructions have multiple actions, several addressing modes (DB-plus, S-minus, Q-plus, Q-minus, P-plus, and P-minus), 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 content of the stack registers as operands. All instructions except the 63 stack operations are in 16-bit format; the stack operations may be packed two per word to further enhance the code density.

A complete set of arithmetic instructions provide 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 II Dispatcher, instructions for enabling/disabling process switching, and instructions for data transfers between data segments contribute greatly to the efficiency of the operating system.

MICROCODE

The entire instruction set of the HP 3000 Series II 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 (formerly software-implemented), the amount of code and execution times are greatly reduced. In addition to the instruction set, other system functions have been microprogrammed, including 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. All of these features are standard in the HP 3000 Series II.

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.

MAIN MEMORY

The significant features of the HP 3000 Series II main memory are listed in table 2-3.

Table 2-3. HP 3000 Series II Main Memory Features

- High-speed, semiconductor, random access memory
- Automatic fault detection and single bit fault correction
- Seven memory sizes ranging from 128K bytes to 512K bytes
- Addressable to 512K bytes
- 90-minutes rechargeable battery packs to maintain memory data during power failure

The HP 3000 Series II uses high-speed, semiconductor, random access memory which provides single-bit fault correction and some double-bit fault detection. The parity system is retained with this feature, thus maintaining integrity over the various busses. Main memory is available in 128K, 192K, 256K, 320K, 384K, 448K, and 512K byte configurations. Due to the modular design of the HP 3000 Series II, any system can be easily expanded from one memory size to another. When there is more than 256K bytes of memory in the system, it is divided into two modules: 256K bytes 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 21 bits — 16 bits of data and five bits for automatic fault detection and correction.

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 user access to input/output devices is by way of the device-independent MPE-II file system. All location of data, buffering, data transfers, and de-blocking are handled automatically by MPE-II. When the user asks to read a named file, he is only implicitly specifying the actual disc address of the file; the file system determines the explicit address for him and performs the read. At another level, when a user asks 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 for the user. Users who must have actual contact with specific devices 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 the user's 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).

Multiplexer Channel. Each Multiplexer Channel handles up to 16 device controllers. The Multiplexer Channel, in conjunction with the IOP, allows the devices connected to it to run concurrently, interleaving their transfers on a word-by-word basis. By multiplexing devices, 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 applied directly to the IOP for transfer to main memory via the Central Data Bus.

Selector Channel. Selector Channel data transfers bypass the IOP completely to provide transfer rates of up to 1.9 million bytes per second for a single device. With the standard system configurations, the Selector Channel can accommodate up to four HP 7905A Cartridge Disc Drives connected to one device controller. Each drive has a storage capacity of 15 megabytes. All Selector Channel data transfers are performed in block mode and the data is applied directly to main memory via the Central Data Bus.

Computer Museum

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 a table known to both hardware and software and it contains, 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 Series II 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: data service priority and interrupt priority. Each of these values is completely independent of the others, and none 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 at 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 higher 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 but a low interrupt priority. Conversely, a device which has a low data rate but has an important interrupt significance (such as an alarm condition) may be configured to a high interrupt priority but a low data service 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 17 internal interrupts (for user errors, system violations, hardware faults, and power fail/restart) plus seven traps for arithmetic errors and illegal use of instructions.

PERIPHERALS

Mass storage devices consist of moving-head disc drives. These units provide storage capacities from 15 to 47 million bytes and data transfers of nearly one megabyte per second. On-line storage can be greatly expanded by adding additional disc drives to the system.

Low cost magnetic tape units are available in 9-channel models with recording densities of 800 or 1600 bytes per inch.

A card reader operates at 600 cards per minute.

An HP 2640A Interactive Display Terminal is supplied as standard equipment for the system console.

Line printers are available with operating speeds of 200, 300, 600, or 1250 lines per minute. The various line printers provide 132-column print lines, using 64, 96, or 128 characters.

High-speed punched tape equipment reads at 500 characters per second. Punched tape output is available as a separate unit operating at 75 characters per second. Paper, plastic, or mylar tape may be used with all units.

An interface is available which allows the user to connect a CalComp Series 500, 600, or 700 (excluding models 745 and 748) Plotter to the system.

A card reader punch provides fully buffered, on-line, 80-column card reading, punching, and printing capabilities in a single device. Off-line data recorder (keypunch) capability is optionally available. The reading rate is 120 cards per minute; the punching/printing rate is 45 to 75 cards per minute, depending upon the number of card columns involved. The punch employs the ANSI Hollerith card code and the printer employs a modified ASCII 64-character set as standard.

Also available are an Asynchronous Terminal Controller and synchronous interfaces for data communications. There may be up to two Asynchronous Terminal Controllers connected to the system, each controlling up to 16 asynchronous terminals.

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

THE OPERATING SYSTEM 3

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

This section discusses the general concepts and capabilities of the operating system itself. To relate the operating system to the overall MPE software environment, compare figure 3-1 to figure 1-1 contained in section I of this manual.

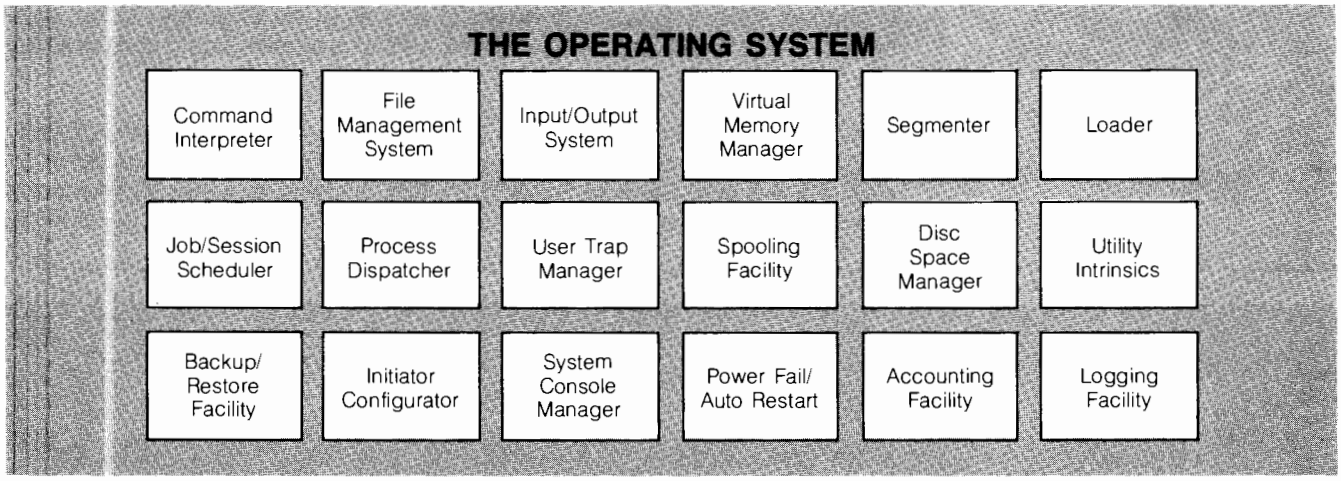


Figure 3-1. HP 3000 Series II Multiprogramming Executive Operating System (MPE)

MULTIPROGRAMMING

One of the major ways in which operating efficiency is achieved in an HP 3000 Series II is by multiprogramming. *Multiprogramming* is a method whereby several related or unrelated programs are in partial states of completion and resources are being allocated among them to balance the system load and to meet response and throughput requirements. While one program is awaiting an occurrence, such as the completion of an input/output operation, control of the central processor is directed to the program which is currently highest in priority of those waiting for the CPU. The controlled competition among several programs for the processing, storage, input/output, and programming facilities of the system helps to ensure that as much of the system as possible is always kept busy performing useful work. As a result, the total throughput of the system (that is, the total volume of work performed by the system during a given interval of time) is significantly increased.

MPE is designed to efficiently and fairly allocate, schedule, and dispatch control of the central processor, storage, and input/output devices among many programs being executed concurrently. MPE, in conjunction with the

architecture of the central processor, also provides complete protection against one program destroying or interfering with another.

The number of programs that can be processed concurrently depends on such factors as the hardware configuration, program operating modes (interactive or batch), and the applications involved. MPE is designed so that the maximum number of concurrently running programs can be expanded or contracted by changing a single system configuration parameter.

In the MPE multiprogramming environment, each programmer uses the computer as though it were his own private machine — in other words, he need not depend on or even be aware of others using the machine. For those cases where programmers need to interact with one another during execution, MPE includes extensive facilities allowing them to do so.

INTERACTIVE PROCESSING

In interactive processing the user enters commands and data through a keyboard terminal and receives immediate responses to his input. This type of interaction, called a *session,* is particularly useful for program development, text editing, data entry, information retrieval, computer-assisted instruction, and many more applications where a direct dialogue with the system can help solve the user's problem. The operating system itself (including access to spooled devices) and all of its language, utility, data base management, and telecommunications programs are accessible by way of sessions.

As an example, consider the case where a user wishes to create a COBOL program and then compile, prepare, and run it. This can easily be done by way of a session. Refer to figure 3-2 which shows the various commands entered by the user during the session. The user initiates the session by

```
:  HELLO   YOURNAME.YOURACCT    Initiates the session.
:  EDITOR   Invokes the HP 3000 Text Editor.
/  SET   FORMAT=COBOL   Specifies that you will be entering COBOL source statements.
/  ADD   Specifies that you wish to create a new text file.
       (COBOL source statements)
/  KEEP   YOURFILE   Saves the text file on disc under the name YOURFILE.
/  EXIT   Terminates the text editor.
:  COBOLGO   YOURFILE   Causes the COBOL source program contained in YOURFILE
                        to be compiled, prepared, and executed.
:  BYE   Terminates the sessions.
```

Prompt characters issued by MPE and the text editor.

Figure 3-2. Sample Session, Command Input Only

pressing the RETURN key on a terminal that is connected on-line to the system. MPE responds by displaying a prompt character (a colon). The user then logs onto MPE by entering a HELLO command containing at least his assigned user and account names. Then he calls the HP 3000 Text Editor and enters two editor commands followed by the COBOL statements that constitute his program. When the entire source program has been entered, he saves it on disc under the file name YOURFILE by entering a KEEP editor command and then terminates the editor subsystem. Now his source program exists as a disc file in the system. To compile, prepare, and execute the program he simply enters a COBOLGO command specifying the file name 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 user's program is finished executing, MPE displays the message END OF PROGRAM followed by another colon prompt character. The user terminates the session by entering the BYE command.

The above example is somewhat simplified in the sense that it does not include the various informational messages, compilation output, and user program output generated by MPE, the text editor, the COBOL Compiler, and the user's program. The fact remains, however, that if the user's 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 user's program.

BATCH PROCESSING

Batch processing lets the user 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, no further information is needed from the user.

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

Several jobs can be submitted to the system from multiple devices concurrently. MPE schedules each job for execution according to its job input priority. Commands are provided for monitoring job selection, and the operator can dynamically adjust the job selection criteria.

MPE executes many sessions and batch jobs simultaneously. The only significant difference between a session and a batch job is that during a session the user can alter the course of processing whereas in a job the command stream is fixed and will be executed in its entirety (assuming that there are no unanticipated errors).

SPOOLING

The MPE spooling facility allows concurrent usage of devices which would otherwise be non-sharable, such as card readers, magnetic tape drives, or line printers. This is accomplished by copying the input or output to a disc where it is processed later. For example, if six users need to produce output on a line printer at approximately the same time, their output is directed instead to spoolfiles on disc from which it 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 to become free. Similarly, if there are ten jobs to be read from a card reader or a 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.

This process of copying input or output to disc as intermediate storage is called *spooling.* The spooling of batch job input can be initiated not only from a card reader or a magnetic tape unit, but also from within an interactive session at the user's terminal by way of the :STREAM command. The :STREAM command temporarily transforms the user's session into an input spooler. Typically the user enters one or more job streams (each beginning with a :JOB command and ending with a :EOJ command) and saves them on disc using the HP 3000 Text Editor. He then issues a :STREAM command whenever he wishes the particular job streams to be executed. In response to the :STREAM command, MPE copies the job streams to spoolfiles and notifies the user of the job numbers assigned to each job stream. The session then continues independent of the spooled jobs.

MPE COMMANDS

The user can cause MPE to perform many system functions through the use of MPE commands. When a session or job is executing, a portion of MPE called the *Command Interpreter* reads each command image from the user's terminal or the batch input device (or spooled disc file), checks its validity, and then causes the appropriate action to take place. After the requested action is successfully completed control always returns to the Command Interpreter which then processes the next command.

The user can interrupt execution of a program at any point in a session by pressing the BREAK key on his terminal (some terminals have an ATTENTION or INTERRUPT key instead of the BREAK key). This returns control to the MPE Command Interpreter, allowing the user to issue commands to abort the program (without disrupting the session), perform various file or utility operations, or resume executing the program. The BREAK key does not, however, interrupt execution of an MPE command.

NOTE: Some MPE commands can also be issued programmatically (that is, from within a user's program). This is done using the COMMAND intrinsic to pass ASCII images of MPE commands to the Command Interpreter.

A colon (:) is the MPE prompt character. When submitting a batch job all MPE commands within that job have a colon in column one; this helps MPE to distinguish the statement as a system command and also aids in detecting the end of data within a batch job stream. During a session, however, MPE automatically displays the colon at the user's terminal to alert the user that MPE is ready to accept another command.

If a command error is detected during a session, MPE responds by displaying an error number and possibly another number specifying which parameter was in error. If the user now enters a carriage return, MPE prompts for another command and the user may continue. If the user enters something other than a carriage return, MPE displays a message describing the command error and then prompts for another command. If a command error is detected during a batch job, the error number and a descriptive message are printed on the job listing device and all card images from that point through the next :EOJ, :DATA, or :JOB command are usually ignored. The :CONTINUE command can, however, be used to cause execution to continue despite a command error.

There are MPE commands for:

- Initiating and terminating jobs or sessions
- Running system programs (such as compilers)
- Running user programs
- Creating, managing, or deleting files
- Displaying file information
- Displaying job and session status
- Displaying device status
- Transmitting messages
- Placing restrictions on user resource consumption
- Debugging
- User accounting of system resource consumption
- System tailoring, documenting, and back-up
- System tuning
- System console operation (job control; device control)
- Spooling
- Logging

Appendix A of this manual gives a complete list of all MPE commands with a brief description of each.

Computer Museum

MPE INTRINSICS

Most system functions are also available to the programmer in the form of special MPE procedures, called *intrinsics,* which are invoked by external calls from the user's program. These external links from user programs to operating system intrinsics are satisfied when the MPE Segmenter prepares the user program for execution. Users may also establish their own "intrinsics" within the system.

System intrinsics are written in the HP 3000 Series II Systems Programming Language (SPL) and follow the rules and constraints of that language. They may usually be called from any COBOL, RPG, BASIC, FORTRAN, or SPL program.

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

Appendix B of this manual gives a complete list of all MPE instrinsics 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 Series II 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 user's program. In such a case, if the user has not specified an appropriate "trap procedure", his batch job is usually removed from the system or his interactive session resumes control at his terminal with a message and a prompt for another command. The :CONTINUE command can, however, be used to continue execution of a batch job despite an irrecoverable intrinsic error.

By initially calling system trap intrinsics, the user may specify special actions to be taken in the event of an irrecoverable error.

THE SYSTEM ADMINISTRATORS

Throughout this manual you will find references to several types of system administrators such as the system manager, the system supervisor, and the account manager. These administrator types are each associated with a particular set of functional capabilities. For example, there is a unique set of MPE commands that can be used only by someone who has been designated as the system manager.

Refer to table 3-1. There is no formal requirement as to how these sets of functional capabilities are divided among the users of a particular HP 3000 System. It is only required that at least one person be designated as each type of administrator. The responsibilities can be divided or shared in virtually any combination desired. In a small installation a single person may be designated as the system manager, the system supervisor, and the account manager for all accounts in the system. In larger installations the responsibilities may be divided among several individuals: one or more system managers, one or more system supervisors, and one or more account managers for each account in the system.

Table 3-1. System Administrator Responsibilities

Administrator	Responsibilities
System Manager	Manages the overall system by creating accounts and defining resource-use limits (if any) for each account.
System Supervisor	Manages the system on a day-by-day basis. Can manage the scheduling queues, alter the system configuration, maintain the system library, and display various items of system information.
Account Manager	Manages one or more accounts by defining the valid users and file groups and specifying resource-use limits (if any) for them.

ACCOUNTING FACILITY

The MPE accounting facility provides user installations with a flexible and powerful means of coordinating access to the system and disc file usage.

Coordinating System Access

The system administrators of an HP 3000 Series II installation can devise a structure of accounts and users which reflects the functional organization of the people who will be using the system. Access to the system is partitioned into accounts, each having a unique *account name.* Individuals who are permitted to initiate jobs and sessions under a particular account name are assigned *user names* within that account. People who will be working together on a project or who will be doing the same type of work (such as entering incoming orders through terminals) are frequently assigned to the same account and could even be assigned the same user name within an account.

When initiating a job or session, the user supplies (as part of the :JOB or :HELLO command, respectively) his user name and account name. To prevent unauthorized access to the system, a password can be associated with each account and user name. Attempts to initiate a job or session which do not include all defined passwords will fail.

MPE maintains running totals on the amounts of system resources that each account consumes. This includes disc space used, cumulative CPU time consumed, and (for sessions) cumulative terminal connect time. The current totals can be displayed at any time and can, of course, be used for billing purposes.

The system manager may establish limits on the amounts of system resources that a particular account can consume. If one of the cumulative time limits is exceeded, no more jobs or sessions can be initiated under that account name until the system manager either adjusts the particular limit or clears the offending total. If the account's disc space limit is exceeded, jobs or sessions running under that account name may not acquire more permanent disc space until either the system manager adjusts the limit or someone within the account frees some disc space.

The system manager can also allocate certain special capabilities to accounts whom he feels are qualified to use them. These include the ability to manipulate the accounting structure, to handle system log files, to create, manipulate, and delete processes, to create extra data segments, and to operate in privileged mode (which allows the account users to access *all* MPE resources, including intrinsics which are otherwise uncallable).

Coordinating File Usage

The overall permanent disc file domain of the HP 3000 Series II is partitioned among the various accounts. Each account's file domain is further partitioned into *groups.*

The fully-qualified name of any file includes the file's name, the name of the group in which it resides, and the name of the account in which it resides. In actual practice, however, the user often does not have to supply all three names to access a file.

When initiating a job or session, the user may specify a group name in addition to his user name and account name. (Each user has a *home group* assigned to him by the account manager. This home group is the default if

he does not specify a group when initiating a job or session.) If he subsequently references a file which resides in both the account and group under which he is running, he may omit the account and group names from the file reference. For example, assume a user is running a session under the account name ACCT1 and the group name GROUP1. In this case he can refer to the fully-qualified name FILE1.GROUP1.ACCT1 as merely FILE1.

The primary advantage of the use of fully-qualified file names (whether specified explicitly or implicitly) is that different files with the same file name can exist in two or more accounts, or in two or more groups within an account, with no conflict. In such a case, the account and group names differentiate the files from one another.

A user may create files only in the account and group under which he is currently running. By using fully-qualified file names, however, he has access to any file present in the system (provided that existing security provisions allow him that access).

The creator of a file may specify various security provisions for that file. These provisions include the type of user allowed to access the file (account members, any users), the types of access allowed (read, write, append, execute, and/or lock), and a file lockword.

THE FILE SYSTEM

All user 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 the user asks to read a named file, he is only implicitly specifying the actual disc address of the file; the MPE File System determines the explicit address for him and performs the read. At another level, when a user asks 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 to the user.

The File System permits sequential access to all files. Disc files with fixed-length records may also be accessed randomly.

Files can be accessed from any of the available programming languages. For the high-level languages, the programmer follows the conventions of the language, not MPE. The particular compiler invokes MPE File System intrinsics as implied by the language constructs. Many programs can simultaneously access the same file; MPE automatically resolves any contention problems which might arise. A single program can access over one-hundred files concurrently, if necessary.

MPE provides two general methods of file security. The first is the use of *passwords.* The creator of a file can establish a password (referred to in the MPE documentation as a *lockword*) which must be correctly supplied whenever anyone makes reference to that file.

The second method of file security is the use of file access mode and user type restrictions. Refer to table 3-2. For each account he creates the system manager specifies the file access modes allowed under that account and the types of users to whom they are available. Similarly, when an account manager creates a group he specifies the file access modes allowed under that group and the types of users to whom they are available. Finally, the creator of a particular file specifies the file access modes allowed for that file and the types of users to whom the file is available. Thus, the total security provisions for any file depend upon specifications made at all three levels — the account, group, and file levels. A user must pass tests at all three levels (the account, group, and file, in that order) to successfully access a file in the requested mode.

Table 3-2. MPE File Access Modes and User Types

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	Allows the user exclusive access to a file if he desires it (if requested in the file reference, no other user can access the file concurrently).

USER TYPES	
Any User	Makes the specified access modes available to any user of the system.
Account User	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.

Commands are provided to :BUILD, :PURGE, and :RENAME disc files. The programmer may specify how disc space is to be dynamically acquired, whether to deal with logical or physical records, whether to include carriage control characters, and so forth. The :RENAME command, which is restricted to the user who created the file, can also be used to establish or change a lockword on a file or to move a file to another group in the account.

The names and locations of all files in the system are maintained in a System File Directory. The :LISTF command can be used to display varying amounts of file directory information about an individual disc file, all files in a particular group, or all files in a particular account.

A temporary disc file set is also available to each job and session. New disc files may be placed in the temporary file set programmatically or by using a

:FILE command. If a temporary disc file is not saved into the account's permanent disc file set, it will be deleted at the end of the job or session. Temporary file space is not included in the account or group file space totals or limits.

Extensive disc file back-up facilities are provided. The :STORE command copies a file, a file set, or several file sets to magnetic tape. The :RESTORE command retrieves the files from magnetic tape. Both multi-file reels and multi-reel files are supported. These two commands, :STORE and :RESTORE, apply for all types of disc files.

The MPE File System includes the most widely used set of system intrinsics on the HP 3000 Series II. With these intrinsics the user can programmatically access and manipulate many files concurrently.

PROCESSES

In an MPE environment, programs are run on the basis of *processes* created and handled by the operating system. The process is the basic executable entity in MPE. A process is not a program itself, but the unique execution of a program by a particular user at a particular time. When a program's execution is requested, 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. Thus, for example, each user working on-line with the BASIC Interpreter is running the interpreter under a separate process. All use the same code (the only copy of the interpreter in the system) but each has his own data stack.

Processes, and the elements that constitute them, are invisible to the user who has only the standard capabilities since he has no direct interest in processes or their structure. For this user, MPE automatically creates, handles, and deletes all processes. The user with process-handling capability, however, can create and manipulate processes directly.

SCHEDULING

Batch job input spooled on disc is selected for execution according to the value of the job input priority parameter (INPRI=x) specified on the :JOB command of each job. The console operator specifies the maximum number of batch jobs that can be executed concurrently by entering a command through the system console. Whenever the system is able to execute another batch job, the input spoolfile with the highest INPRI value is selected for execution. If there are two or more input spoolfiles with the same INPRI value, the oldest one is selected first.

Spooled output on disc is selected for processing according to the value of the output priority parameter (OUTPRI=x) specified on the :JOB or :HELLO command of the associated job or session. When the system is able to process another output spoolfile, the one with the highest OUTPRI value is selected. If there are two or more output spoolfiles with the same OUTPRI value, the oldest one is selected first.

The processes which constitute the currently running jobs and sessions are granted control of the CPU by the MPE Dispatcher. The user may select a general category of process dispatching priority by including the PRI= parameter in his :JOB or :HELLO command. The four process dispatching priority types available to users are:

BS - very high priority
CS - timeshare
DS - batch
ED - very low priority (background)

The MPE Dispatcher translates these priority types into numerical ranges within one overall scheduling queue. The numerical range of each priority type can be changed while the system is running by entering an MPE command. This powerful capability makes it possible for the console operator or the system administrators to dynamically tune the system so that an optimal balance of service is maintained between jobs and sessions at any given time.

THE SEGMENTER

Three steps are involved in taking a program from source form to an executable state. The first is to compile the source program into *Relocatable Binary Modules* (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* (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 is done by the MPE Segmenter. The final step is for the MPE Loader to 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. At this point the segments are part of active virtual memory, all external references have been satisfied, and the program is scheduled for execution. Often all of these steps are initiated by a single MPE command (as an example, refer to the session illustrated in figure 3-2). When necessary, however, the user can initiate each step individually, controlling what happens along the way.

The user initiates compilation, preparation, and allocation/execution of programs with the following set of MPE commands:

Prepare	:PREP	
Execute	:RUN	
Prepare and Execute		:PREPRUN
COBOL	:COBOL	(compile only)
	:COBOLPREP	(compile and prepare)
	:COBOLGO	(compile, prepare, and execute)
RPG	:RPG	(compile only)
	:RPGPREP	(compile and prepare)
	:RPGGO	(compile, prepare, and execute)
FORTRAN	:FORTRAN	(compile only)
	:FORTPREP	(compile and prepare)
	:FORTGO	(compile, prepare, and execute)
BASIC Compiler	:BASICOMP	(compile only)
	:BASICPREP	(compile and prepare)
	:BASICGO	(compile, prepare, and execute)
SPL	:SPL	(compile only)
	:SPLPREP	(compile and prepare)
	:SPLGO	(compile, prepare, and execute)

Program preparation is performed by the Segmenter in response to a :PREP command or one of the combining forms which includes PREP or GO. Many users will not need the Segmenter for anything more than that.

Some users, however, may wish to manage USL files by adding, deleting, activating, or deactivating RBMs within them or may wish to build and manage Segmented Libraries (SLs) which are used to resolve external references from user programs. This can be done by invoking the Segmenter and using its set of commands.

Occasionally a user may wish to alter the segmentation of a program to improve its run time efficiency. In many systems he would have to recompile his program in order to do this. With MPE, however, the segmentation can be modified by simply using the Segmenter to rearrange the RBMs and then preparing the USL file into a new program file.

A convenience feature of the Segmenter is the ability to have different versions of the same subprogram within a single USL file. The Segmenter commands have an optional index parameter which specifies the "i'th" most recent definition of an entry point within a particular subprogram. By using this optional index parameter the user has complete control over the activating and deactivating of entry points within various versions of his subprograms.

CAPABILITY SETS

A typical HP 3000 Series II installation has a large variety of users with a wide range of skills and needs. To protect the system and its users, the system administrators can restrict certain system capabilities to only those users who fully understand their correct use. This is done through the use of *capability sets*. When the system manager creates an account, he defines for it a capability set that determines whether or not the users of that account are allowed to perform such system functions as manipulating the accounting structure, handling system log files, creating, manipulating, and deleting processes, and creating extra data segments. When an account manager defines the users of his account, he defines a capability set for each. A user's capability set may consist of any or all of the capabilities available to the account. The account managers also define a capability set for each group in their account. Group capability sets limit the capabilities available to any program residing in a file within that group.

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

THE LOGGING FACILITY

The MPE logging facility optionally monitors the use of system resources. Details about system resource requests are recorded in a series of *Log Files* on disc. The system supervisor of each installation selects those system and user events that are to be recorded in the Log Files.

The Log Files are ideal sources of information for billing purposes. Each log record is correlated to the job or session that caused the event. There are log records for job or session initiation or termination, program termination, file closing, file spooling completion, and system shut-down. The contents of the Log Files — the individual log records — are not displayed or otherwise used by MPE. The installation may write various analysis routines for this purpose. Details contained in each log record permit an installation's accounting programs to bill accurately for system usage. This is a problem solver in any installation where resources are shared by fiscally independent users.

I/O device failures on any device are also recorded in the Log Files. These log records are then available for detecting problems with devices before they begin to interfere sigificantly with the overall system operation. Note that I/O failures for disc and magnetic tape are automatically retried several times by the operating system and each retry (in addition to the original failure) is recorded in the Log Files.

START-UP AND MODIFICATION OF MPE

MPE is initially brought up by the console operator through a simple restart operation. There are five restart modes:

- WARMSTART — the system is restarted from the disc. Spoolfiles are recovered and incompletely processed spooled jobs can be automatically restarted.
- COOLSTART — the system is restarted from the disc. The previous spooling environment at :SHUTDOWN time is not restored, but resident user files are saved.
- COLDSTART — the system is read from magnetic tape. The I/O configuration and system configuration from the tape are used to define the system. These are merged with the directory and files present on the disc.
- UPDATE — similar to cold start, except that the I/O configuration and system configuration on the disc are used to define the system. This mode would be used when starting the system with an updated MPE tape from HP.
- RELOAD — the entire system, directory, files, and configuration information are read from magnetic tape.

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.

Computer Museum

SYSTEM BACK-UP AND RECOVERY

The :SYSDUMP command, which is available to the system supervisor, is used to back-up the MPE system and user files on magnetic tape. These tapes may then be used to reload the system following a hardware or software failure or to transfer the system to another hardware installation.

If the system supervisor is using :SYSDUMP to create a tape for reloading on a different or changed HP 3000 Series II, he may specify a new I/O configuration and system table configuration. In this case :SYSDUMP initiates an interactive dialogue which allows him to specify the desired changes. There are eight possible areas of change, any or all of which may be skipped. These areas are:

- I/O devices
- System table sizes
- System disc allocation
- Logging facility
- Miscellaneous changes
- Scheduling changes
- Segment size limits
- System modules to be included.

The dialogue also allows the system supervisor to define a *back-up date*. All disc files altered on or after the supplied date will be copied to the back-up tape(s). This facility can be used to back-up specifically those disc files which change each day.

POWER FAIL/ AUTO RESTART

When an AC power failure is detected by the HP 3000 Series II, 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 actually in use when the power failure occurred).

PROGRAMMING LANGUAGES 4

This section summarizes the various programming languages that are available to the user of an HP 3000 Series II. To relate these languages to the overall MPE software environment, compare figure 4-1 to figure 1-1 contained in section I of this manual.

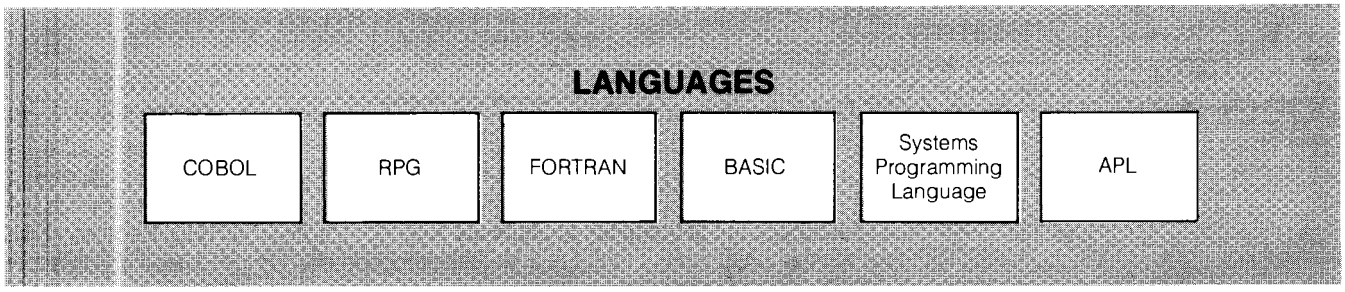


Figure 4-1. HP 3000 Series II Programming Languages

MULTILINGUAL PROGRAMMING

The HP 3000 Series II provides five programming languages to meet the needs of a wide range of users:

- COBOL — A business programming language whose syntax is quite similar to ordinary English.
- RPG — A business programming language designed primarily for complex report generation.
- FORTRAN — A programming language originally designed for execution of scientific formulae. However, it is now used extensively in business as well as the scientific community.
- BASIC (both an Interpreter and a Compiler) — A programming language designed to simplify "on line" terminal usage. It is now used extensively in business, educational, and scientific environments.
- SPL — A programming language unique to the HP 3000 Series II which combines the best features of "high level" and assembly languages. It is a block-structured language similar to ALGOL.
- APL — A terminal-oriented, high level language, free of detail and with a large set of functions and operators.

These languages may be used concurrently in any combination by many users or for multiple compilations in any combination within a single user's job or session.

With the exception of the BASIC Interpreter, the user invokes all language compilers on an HP 3000 Series II in the same general way. This reduces the necessary training, since learning to use one compiler is essentially learning to use them all. The HP 3000 Series II hardware preserves the user's programming investment in that subroutine call conventions are in the machine's microcode. This enables programs in all languages to call subroutines written earlier in other languages. A user's program written in COBOL, RPG, FORTRAN, BASIC (interpreted or compiled), SPL, or APL can call subroutines written in COBOL, FORTRAN, BASIC (compiled), or SPL.

A user's program written in any of the available programming languages may manipulate an IMAGE data base (see section VI of this manual). This brings the HP 3000 Series II data base management capabilities to wherever they are needed.

All programs in an HP 3000 Series II use the MPE File System. This single File System provides a logical data space to user programs written in any language. By using MPE File System intrinsics, the user's program manipulates data in this space. In so far as the user's program is concerned, these data accesses are independent of the physical type of devices on which the data resides. For example, no changes need be made to a user's program to switch input or output from cards to an interactive terminal or from magnetic tape to disc. Programs used in batch jobs can also be used without modification in interactive sessions, and vice versa.

COBOL/3000

COBOL, the COmmon Business Oriented Language, is a "high level" programming language developed for use in business environments. Like all "high level" languages it is problem oriented and relatively free of hardware constraints. The COBOL language consists of a set of statements whose appearance is very similar to ordinary English phrases and sentences. This allows the programmer to focus his attention on the logical steps involved in solving his programming problem rather than on the detailed actions which must occur within the computer. For example, to add one value to another the COBOL programmer uses a statement such as

```
ADD BONUS TO SALARY
```

instead of writing the specific machine instructions which reference actual memory addresses and manipulate the computer's storage registers. This English-like syntax also means that COBOL programs are very readable and are essentially self-documenting.

The standard for this language is American National Standard COBOL X3.23-1968 as approved by the American National Standards Institute. American National Standard COBOL includes eight processing modules:

- Nucleus
- Table Handling
- Sequential Access
- Random Access
- Sort
- Segmentation
- Library
- Report Writer

COBOL/3000 is an implementation of all of these modules except the Report Writer. All COBOL/3000 modules except the Report Writer have a "high" ANSI rating.

Figure 4-2 shows a sample COBOL source program.

COBOL/3000 is described in the *COBOL/3000 Compiler Reference Manual* (HP part number 32213-90001). As suggested by its title, that manual is strictly a reference manual for the HP 3000 version of COBOL. Programmers who desire a general tutorial text are referred to *A Guide for COBOL Programming* by Daniel D. McCracken and Umberto Carbassi (2nd Edition, New York, Wiley Interscience 1970).

```
IDENTIFICATION DIVISION.
PROGRAM-ID. COBOL-TEST1.
AUTHOR. YOUR NAME.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
77  EDIT-FIELD      PIC $Z,ZZ9.99.
77  TOTAL-COST      PIC 999V99.
77  COST-OF-SALE    PIC 99V99.
77  TAX             PIC 99V99.
77  Y-N             PIC X.

PROCEDURE DIVISION.
ENTER-ROUTINE.
    MOVE ZEROS TO TOTAL-COST.
    DISPLAY SPACE.
    DISPLAY "ENTER COST OF SALE (BEFORE TAX) NO DECIMAL PT".
    DISPLAY "(4 DIGITS MAX) INCLUDE LEADING ZEROS!".
    ACCEPT COST-OF-SALE.
    COMPUTE TAX = COST-OF-SALE * .06.
    ADD COST-OF-SALE, TAX TO TOTAL-COST.
    MOVE TOTAL-COST TO EDIT-FIELD.
    DISPLAY "TOTAL COST OF PURCHASE = " EDIT-FIELD.
    DISPLAY "ARE YOU FINISHED? (Y OR N)".
    ACCEPT Y-N.
    IF Y-N = "N" GO TO ENTER-ROUTINE.
    STOP RUN.
```

Figure 4-2. Sample COBOL Source Program

Hewlett-Packard also offers the *3000 COBOL Self Study Course* (HP part number 22957-90000) which is distributed on magnetic tape cassettes.

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.

RPG programs can handle almost any business data processing task. For instance, they can:

- Process large tables and arrays of data
- Read data from input files on various devices
- Perform extensive calculations and store the results in new files on discs, tapes, terminals, or punched cards
- Update large files on magnetic tape or disc
- Generate multiple reports of varying complexity in a single process

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 the programmer to specify many important operations with a minimum of effort: he merely makes simple entries on specially-formatted coding sheets (refer to figure 4-3). 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.

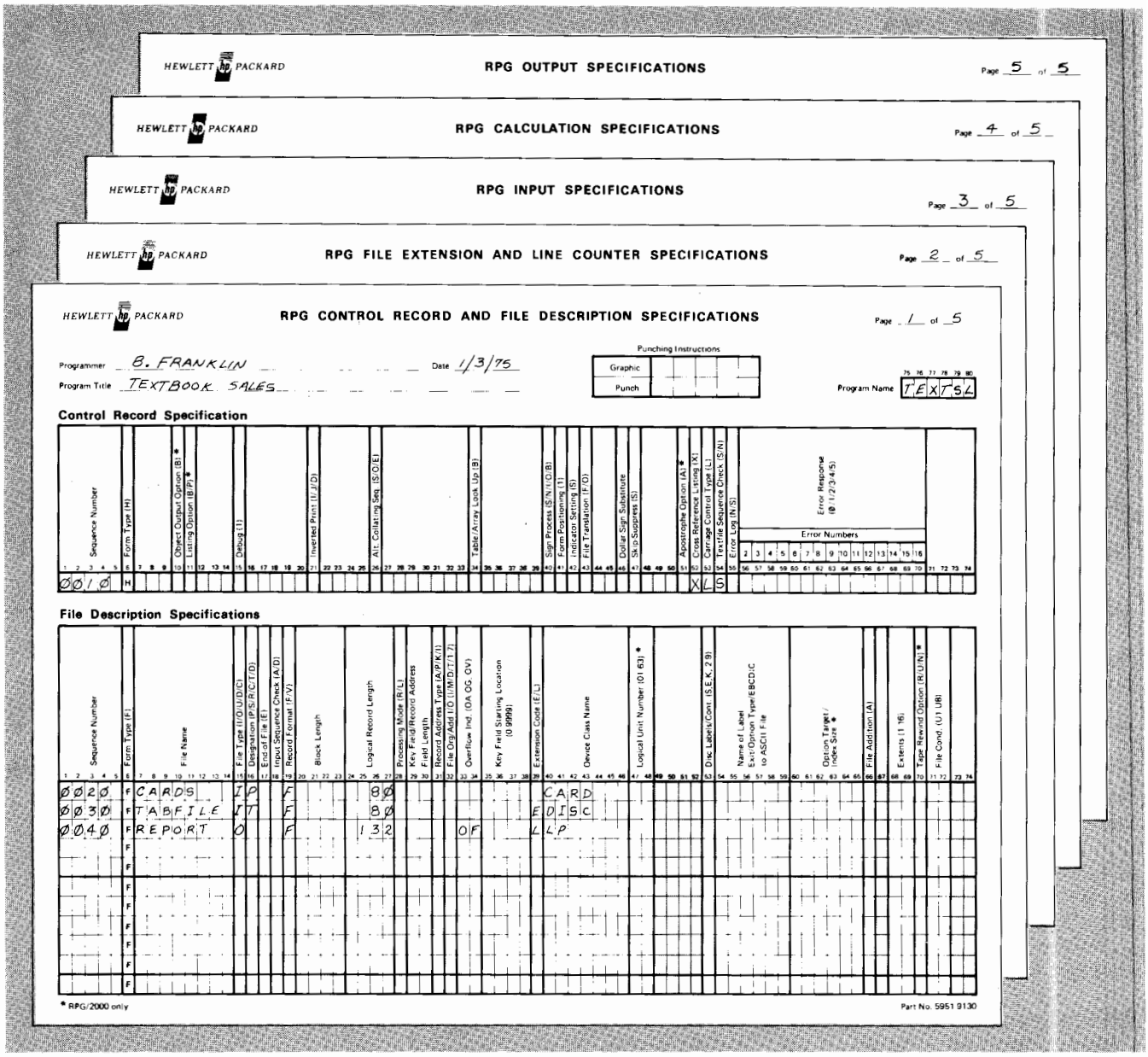


Figure 4-3. Sample RPG Source Program

RPG/3000 is described in the *RPG/3000 Reference and Application Manual* (HP part number 32104-90001). As suggested by its title, that manual is primarily a reference manual for the HP 3000 version of RPG. Programmers who desire a general tutorial text are referred to:

Bernard, Solomon Martin, *System/360 Report Program Generator*. New Jersey, Prentice-Hall, Inc., 1970.

Brightman, Richard W. and Clark, John R., *RPG I and RPG II, System/3 and System/360*. Ontario, The Macmillan Company, Collier-Macmillan Canada, Ltd., 1970.

The *IBM System/3 to HP 3000 Conversion Guide* (HP part number 32104-90004) tells how to modify RPG programs written for an IBM System/3 so that they can be run on the HP 3000 Series II.

The *RPG Listing Analyzer* (HP part number 32104-90003) is a pocket-sized template which facilitates the analyzing of RPG program listings.

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.

Like all "high level" programming languages FORTRAN is problem oriented and relatively free of hardware constraints. The language consists of a set of statements whose appearance is often similar to mathematical formulae. This allows the programmer to focus his attention on the logical steps involved in solving his programming problem rather than on the detailed actions which must occur within the computer. For example, to add two values together, divide the sum by two, and store the result in a named location the FORTRAN programmer uses a statement such as

```
RESULT = (A+B)/2
```

instead of writing the specific machine instructions which reference actual memory locations and manipulate the computer's storage registers.

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).
- A label may be used as an actual argument in a CALL statement to allow alternative return points following execution of the subroutine referenced by CALL.
- STOP or PAUSE statements use a decimal integer or character string for identification rather than an octal integer.
- Both 16-bit and 32-bit integers are supported.
- Subroutines and functions may have secondary entry points.
- A built-in cross reference facility is available as a compile-time option.
- Symbolic debugging is available through the TRACE/3000 facility.
- Undefined variables are detected at compile time.
- Generic functions are recognized.

Figure 4-4 shows a sample FORTRAN source program.

```
C FORTRAN/3000 EXAMPLE
      DO 10 I=1,3
      B=I
      A=SIN(B)+COS(B)
      WRITE(6,15)A
15    FORMAT(" VALUE OF A= ",F5.2)
10    CONTINUE
      STOP "PROGRAM COMPLETED"
      END
```

Figure 4-4. Sample FORTRAN Source Program

FORTRAN/3000 is described in the *FORTRAN Reference Manual* (HP part number 30000-90040). As suggested by its title, that manual is primarily a reference manual for the HP 3000 version of FORTRAN. Programmers who desire a general tutorial text are referred to *A Guide to FORTRAN IV Programming* by Daniel D. McCracken (Second Corrected Printing, December 1967, John Wiley & Sons, New York, London, Sydney).

The *IBM 1130/1800 to HP 3000 FORTRAN Conversion Guide* (HP part number 36995-90013) tells how to modify FORTRAN programs written for an IBM 1130 or 1800 so that they can be run on the HP 3000 Series II.

BASIC/3000

BASIC, the Beginner's All-purpose Symbolic Instruction Code, is a "high level" programming language developed at Dartmouth College in 1963-64 for use in an educational environment. 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.

BASIC is a language whose simple constructs and format make it ideal for on-line programming at interactive terminals. The BASIC/3000 Interpreter was designed primarily for on-line use, but the MPE file facility makes it usable also in batch mode.

The BASIC/3000 Interpreter provides the user with the most powerful version of BASIC available. In addition to all features commonly found in BASIC interpreters throughout the industry, it contains the following extended capabilities:

- Four numeric data types (real, integer, complex, extended precision).
- Mixed-mode arithmetic.
- Multiple-line statements and functions.
- Picture output formats.
- Program chaining with common storage.
- Timed input (by way of the ENTER statement).
- External routine calls.
- Strings and string arrays.
- Direct and sequential access to files.
- File creation and purging under program control.
- User-definable file security, with passwords.
- Trace-debug facility.

- Optional treatment of programs as procedures (through the INVOKE statement).
- IF-THEN-ELSE statements and DO blocks.

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.

Figure 4-5 shows a sample BASIC Interpreter program.

```
 10 FOR I=1 TO 3
 20   PRINT "ENTER VALUE, LESS THAN 100"
 30   INPUT A
 40   IF A>=100 THEN 100
 50   READ B
 60   C=SIN(A)+COS(B)
 70   PRINT USING 120;C
 80 NEXT I
 90 STOP
100 PRINT "BAD VALUE. TRY AGAIN."
110 GOTO 30
120 IMAGE "ANSWER IS:",6D.DD
130 DATA 33.3,50,66.7
```

Figure 4-5. Sample BASIC Interpreter Program

The HP 3000 version of *BASIC is described in the BASIC Interpreter Reference Manual* (HP part number 30000-90026). The use of the BASIC/3000 Compiler is described in the *BASIC/3000 Compiler Reference Manual* (HP part number 32103-90001). As suggested by their titles, those manuals are both primarily reference manuals. Programmers who desire a general tutorial text are referred to:

BASIC Programming by John G. Kemeny and Thomas E.Kurtz (Second Edition, 1971, John Wiley & Sons, New York, London, Sydney, Toronto)

BASIC, Edited by Stephen V.F. Waite and Diane G. Mather (Sixth Edition, 1972, University Press of New England, Hanover, New Hampshire 03755)

BASIC for Beginners (HP part number 03000-90025)

The BASIC/3000 language is summarized in the *BASIC/3000 Interpreter Pocket Guide* (HP part number 03000-90050).

SPL/3000

SPL, the Systems Programming Language, is a new language designed by Hewlett-Packard especially for writing systems software for the HP 3000 Series II. 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.

Traditionally, "high level" (compiler and interpreter) languages and machine-dependent languages have been regarded as mutually-exclusive groups of languages. "High level" languages isolate the programmer from the intricacies of the machine and simplify writing programs by allowing him to write statements that closely resemble those of the spoken language or those of the language used in mathematical formulae. They do not, however, generate object code as efficiently as machine-dependent languages nor do they provide the same precise control at the machine level. Machine-dependent languages, on the other hand, usually offer a syntax that closely matches the structure of the machine-language itself. Although they provide great efficiency and control, they are difficult and time consuming to use because of their complexity and intricate detail and because they place the burden of many supporting functions directly upon the user.

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.

"High Level" SPL Features

SPL provides many features found only in "high level" programming languages such as PL/I or ALGOL. It provides a powerful, "high level"syntax structure similar to that of ALGOL. In fact, anyone who knows the ALGOL language can begin programming in SPL almost immediately; others will find it much easier to learn than a conventional assembler language. To provide greater compatibility with the HP 3000 Series II and satisfy other systems programming needs, SPL incorporates certain features that extend beyond or differ from those of ALGOL.

SPL allows the user to write simple, compound, and nested "high level" statements; procedure declarations and calls; recursive procedures and subroutines; and assignment directives, all in free-form structure. Programs are delimited by BEGIN-END statement pairs and consist of other statements and declarations, all separated by semicolons. Declarations relate names to various elements within the program — variables, arrays, labels, pointers, subroutines, and procedures, for instance. Statements perform operations on these elements or transfer control within the program. SPL permits manipulation of integer, double integer, packed decimal, real, long real, logical, bit, and byte quantities. Typical statements are IF-THEN-ELSE, FOR, DO-UNTIL, WHILE-DO, CASE, and GOTO. These simple constructs not only make SPL programs easy to write, but also make the program listings easy to interpret.

Machine-Dependent SPL Features

Although the SPL language processor is a compiler rather than an assembler, it offers many features that are tied explicitly to the HP 3000 Series II

hardware structure and grant the programmer the exact control and efficiency he requires to develop system software. Among these features are:

- An ASSEMBLE statement, providing a built-in assembler that generates sequences of machine instructions.
- Explicit addressing of hardware registers.
- Extraction, depositing, and shifting of bit fields.
- Statements that appear as high-level constructs but are implemented in one machine instruction each. These are the SCAN, MOVE, PUSH, SET and DELETE statements.
- Branching based upon hardware status.
- Use of all operators, data types, and other features inherent in the hardware.

Because every syntax structure in SPL is simply and directly related to some feature of the hardware instruction set, the SPL/3000 Compiler can generate machine code very efficiently. Because SPL allows the programmer to explicitly use all hardware addressing modes, machine instructions, and registers, it is ideal for systems software development.

Sample SPL Procedure

The SPL procedure shown in figure 4-6 scans an input buffer containing a specified number of words, searching for one of five special characters. The procedure skips initial blanks and moves subsequent characters to an output buffer. It returns the special characters found, the length of the output buffer, and the next character of the input buffer that was not scanned.

Name of procedure: SCANR

Parameters:	IN	input buffer
	LENGTH	length (in characters) of IN
	OUT	output buffer
	COUNT	number of characters moved to OUT
	BREAK	the special character that was found
	NEXT	next character of IN not scanned

The following definitions of statement elements will help the reader understand the procedure.

Element	Meaning
<< >>	A pair of broken brackets is used to enclose comments.
X	The character X refers to the Index Register.
"..."	Quotes enclose character strings.
;	Statements are terminated by semicolons.
:=	This symbol means "is assigned the value."
TOS	This operator refers to the value of the Top of Stack.
@	This sign means either 1) operate with the contents of a pointer, not the item pointed to, or 2) operate on the address of a variable, not the contents of the variable.

Element	Meaning
POINTER	A pointer is a quantity used to point at another item (provide one-level indirect addressing).
LSL	This instruction requests a logical shift left.
DXBZ	This instruction decrements X, branches if zero.
DEL	This instruction deletes the top item on the stack.
INCX	This instruction increments the Index Register.
DZRO	This instruction pushes a double zero onto the stack.
*	The asterisk, used in the SCAN statement, means that parameters are on the stack.

```
PROCEDURE SCANR(IN,LENGTH,OUT,COUNT,BREAK,NEXT);
  VALUE LENGTH,IN,OUT;
  LOGICAL NEXT;                          <<ADR NEXT UNPROCESSED CHAR>>
  INTEGER LENGTH,BREAK,COUNT;            <<INPT-LGTH,BRK-CHAR,CHARS-MOVED>>
  BYTE POINTER IN,OUT;                   <<INPUT,OUTPUT BUFFER POINTERS>>
<<END OF PROCEDURE HEAD>>
BEGIN
    INTEGER X=X;                         <<INDEX REGISTER>>
    LOGICAL A:="#S", B:="%?", C:=%26000;     << #S%?,<NULL> BRK TABL>>
    LABEL L0,L1,L2;
<<END OF LOCAL DECLARATIONS>>
    X:=LENGTH;                           <<SET INDEX REG TO BUFFER LGTH>>
    ASSEMBLE(INCX,DZRO);                 <<INCREMENT BUFFER LGTH>>
    BREAK:=TOS;                          <<SET BREAK CHAR TO ZERO>>
    COUNT:=TOS;                          <<SET MOVED CHAR COUNT TO ZERO>>
L2: IF DXBZ THEN                         <<END OF STRING CHECK>>
    GOTO L0;                             <<LENGTH EXHAUSTED>>
    IF IN=" " THEN                       <<SKIP LEADING BLANKS>>
      BEGIN
        @IN:=@IN+1;                      <<INCREMENT INPUT POINTER>>
        GOTO L2;
      END;
L1: TOS:=@A & LSL(1);                    <<SET UP BREAK TABLE ADDRESS>>
    SCAN * UNTIL IN,0;                   <<SCAN BREAK TABLE>>
    IF CARRY THEN
      BEGIN                              <<BREAK NOT FOUND>>
        OUT:=TOS;                        <<MOVE CHAR TO OUTPUT>>
        @OUT:=@OUT+1;                    <<INCREMENT OUTPUT POINTER>>
        COUNT:=COUNT+1;                  <<INCREMENT MOVED CHAR COUNT>>
        @IN:=@IN+1;                      <<INCREMENT INPUT POINTER>>
        ASSEMBLE(DEL);                   <<DELETE SCAN ADDRESS>>
        IF DXBZ THEN                     <<END OF STRING CHECK>>
        GOTO L0;
        GOTO L1;
      END;
    BREAK:=TOS;                          <<BREAK FOUND - RETURN THE CHAR>>
    @IN:=@IN+1;                          <<INCREMENT INPUT POINTER>>
L0: NEXT:=@IN;                           <<RETURN NEXT CHAR ADDRESS>>
END  <<SCANR>>;
```

Figure 4-6. Sample SPL Source Program

Documentation

SPL is described in the *Systems Programming Language Reference Manual* (HP part number 30000-90024) and the *Systems Programming Language Textbook* (HP part number 30000-90025). The language is summarized in the *SPL/3000 Programmer's Pocket Guide* (HP part number 5951-4426).

APL

APL, A Programming Language, is a unique "high" level programming language developed by Dr. Kenneth Iverson for scientific and mathematical applications. In recent years it has been used more exclusively in such commercial areas as statistics, modeling, and finance. With APL, the user has the ability to express complex calculations in a simple and concise manner, which allows the user to put numerous computer operations into a few lines of code. As a result of this power, difficult programs can be written in much less time than with other high level languages.

APL Features

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

- Virtual Workspaces — Workspace size in the past has severely limited the use of APL in some applications. With APL\3000, a virtual memory scheme is used which allows extremely large virtual workspaces.

- Dynamic Compiler — APL has always been implemented as an interpreter. APL\3000, on the other hand, actually compiles and saves the code necessary to execute an APL function. When the function is called again, the compiled code is used (after first checking to see if the code is still usable on the current data types and ranks) which means that there can be a significant speed-up of execution on subsequent runs of a function.

- APLGOL — A structured programming extension to APL has been added, which uses ALGOL-like control structures, with the power of APL operators.

  In APLGOL, key words that describe the flow of a given procedure are incorporated with the standard APL expressions, while APL-only programs provide a branch function to control the flow through the algorithm. 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, the user can use disc files, tape files, card readers and other devices. As a part of this facility, it is also possible to issue MPE commands such as file equations which increase the flexibility of the entire APL subsystem.

- Extended Control Functions — A set of functions is provided in APL\3000, which allows the user to organize the relationship between user-defined functions in a flexible manner, obtain access to local variables and branch to labels in other functions.

- Powerful 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. An editing capability is also available in calculator mode to save retyping an entire complicated line when only simple typing errors need correction.

APL Character Set

The APL character set consists of alphabetic characters, underscored alphabetic characters, numeric characters, the blank character, and special characters (graphic symbols). The complete set of APL\3000 characters is shown in figure 4-7.

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

0 1 2 3 4 5 6 7 8 9

¨	dieresis	¯	overbar	<	less
≤	not greater	=	equal	≥	not less
>	greater	≠	not equal	∨	or
∧	and	+	plus	-	bar
×	times	÷	divide	?	query
ω	omega	∊	epsilon	ρ	rho
~	tilde	↑	up (arrow)	↓	down (arrow)
⍳	iota	○	circle	*	star
←	left (arrow)	→	right (arrow)	α	alpha
⌈	upstile	⌊	downstile	_	underbar
∇	del	∆	delta	∘	null
⎕	quad	[	open bracket	]	close bracket
⊂	open shoe	⊃	close shoe	∩	cap
∪	cup	⊥	base	⊤	top
|	stile	/	slash	\	slope
'	quote	(	open parenthesis	)	close parenthesis
;	semicolon	:	colon	,	comma
.	dot		space		
$	dollar	⊢	left tack	⊣	right tack
{	open bracket	}	close bracket	◇	diamond

The following characters are formed by overstriking

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

⍙	delta under	⍒	del stile	⍋	delta stile
⍟	log	⍜	circle bar	⍉	circle slope
⌽	circle stile	⌿	slash bar	⍀	slope bar
⍲	nand	⍱	nor	⍫	del tilde
⍮	base null	⍭	top null	⍬	cap null
⍞	quote dot	⌶	I-beam	⍈	quote quad
⍇	domino				

Figure 4-7. APL\3000 Character Set

APL Keyboard

APL programs generally are composed and executed using terminal devices having special APL keyboards. The Hewlett-Packard HP 2641A Terminals provide such a keyboard. The keyboard for the HP 2641A is shown in figure 4-8. APL\3000 also permits the use of standard ASCII terminals to create and run programs. These terminals, of course, do not have the special APL character set and therefore two or more keys must be struck to form the desired APL character.

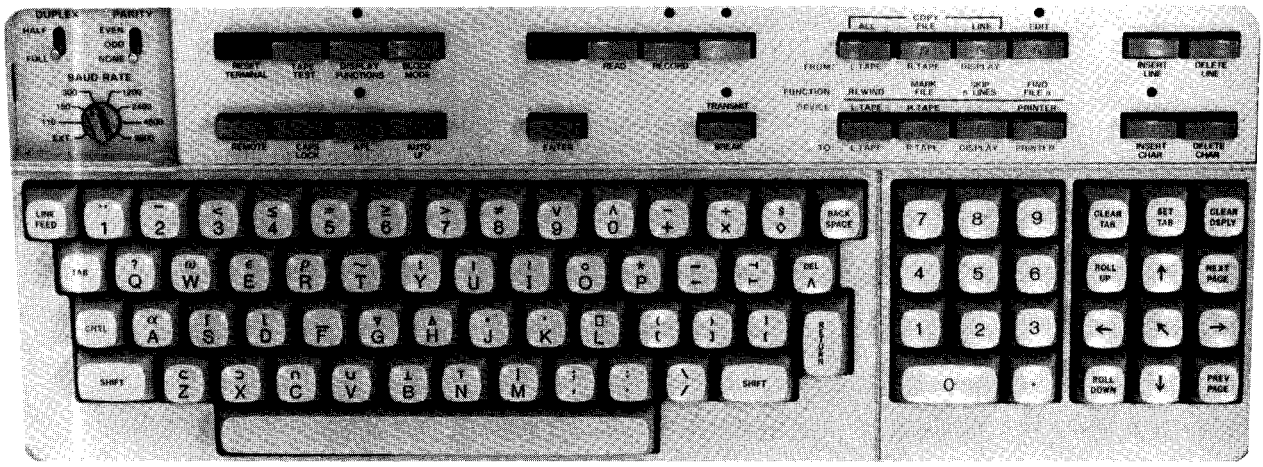


Figure 4-8. APL\3000 Keyboard

Calculator Mode vs. Function Mode

Once a session is initiated, APL can be run in either of two modes: calculator or immediate execution mode, or function definition mode.

In calculator mode, expressions are created and the results may be displayed on the terminal immediately after entering a carriage return.

For example,

```
      6+7
13
      6+7-9
4
      6+7x(56-45)
126
      2 4 8+3
5 7 9 11
```

Assign a variable to the variable A:

```
      A 14+98.5
```

If the variable A is entered, APL displays its value:

```
      A
112.5
```

In function definition mode, the APL editor is used to form expressions into user-defined functions for later use. These user-defined functions formed with the editor may then be invoked from calculator mode or from within another function to perform a computation.

For example, CIRCLEAREA is a user-defined function to compute the areas of sectors of circles.

```
[0]    AREA RADIUS CIRCLEAREA DEGREES
[1]    AREA (oRADIUS*2)xDEGREES 360
```

RADIUS and DEGREES are arguments of the function CIRCLEAREA. RADIUS denotes the radius of the circle and DEGREES denotes the angle the sector subtends.

To run this user-defined function, the name is entered with the appropriate arguments:

```
      163.2 CIRCLEAREA 37.4
8692.791899
```

The HP 3000 version of APL is described in the APL Reference Manual (HP part number 32105-90002) and is summarized in the APL Pocket Guide (HP part number 32105-90003).

UTILITIES

Computer Museum

5

The HP 3000 Series II includes a set of utility programs and subroutine libraries that extends the capabilities of the operating system and language compilers. To relate these utilities to the overall MPE software environment, compare figure 5-1 to figure 1-1 contained in section I of this manual.

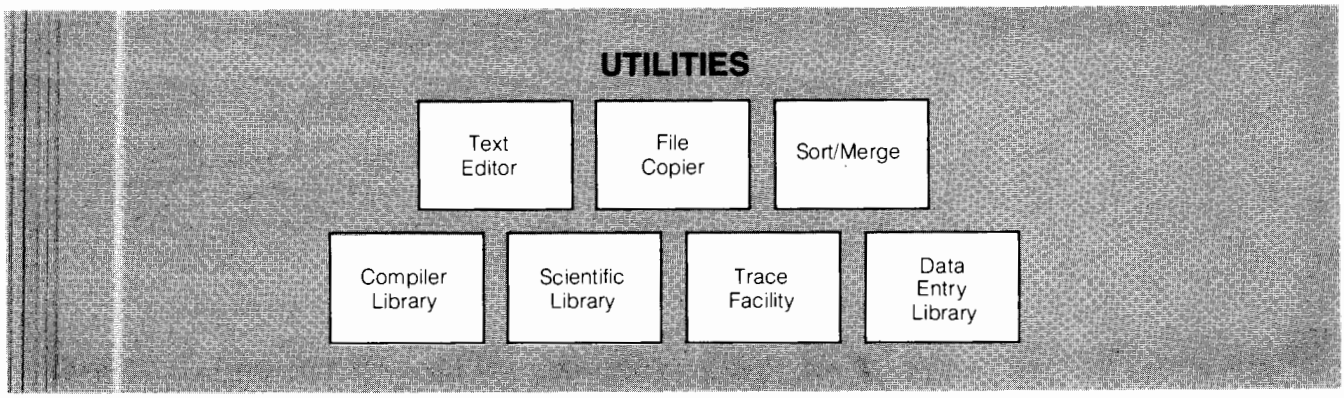


Figure 5-1. HP 3000 Series II Utilities

TEXT EDITOR

EDIT, the HP 3000 text editor, allows the user to create and manipulate files of upper- and lower-case ASCII characters with great ease. Lines, strings, and individual characters can be searched for, inserted, deleted, and replaced. The files to be edited may contain source language programs (COBOL, RPG, FORTRAN, BASIC, or SPL) or text material such as reports.

The user interacts with EDIT through a set of editor commands that includes commands common to many other text editors throughout the computer industry. This makes it relatively easy for users with previous editing experience to learn to use EDIT/3000. The various files used by EDIT are summarized in table 5-1 and all of the available editor commands are summarized in tables 5-2 and 5-3.

In addition, EDIT has three powerful features not usually encountered in other text editors. They are discussed in the following three paragraphs.

EDIT lets the advanced user 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, the user can store sequences of editor commands in the system and then invoke them at will to act upon text data in the editor's work file. In this manner commonly-used sequences of editor commands, such as complex WHILE sequences, can be activated by way of a single USE command rather than by reentering the entire sequence.

A very powerful feature of EDIT/3000 is that it allows the user to write procedures in COBOL, FORTRAN, or SPL that perform editing functions not

otherwise possible with the standard repertoire of editor commands. The user writes, prepares, and stores his procedures on disc and then invokes 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 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 the user to perform any editing function that can be done a line at a time.

The HP 3000 text editor, which can be run in either a batch job or an interactive session, is described in the *EDIT/3000 Reference Manual* (HP part number 03000-90012).

Table 5-1. EDIT/3000 Files

File Name	Function
Input File	The input file is normally used to enter editor commands and text records. Generally, this file is a terminal in an interactive session or a batch input device (such as a card reader) in a batch job.
Output File	EDIT normally transmits messages (and prompt characters in an interactive session) to the output file. Generally, this file is a terminal in an interactive session or a line printer in a batch job.
Work File	The work file contains the text records to be modified. To minimize the possibility of a user accidentally destroying his only copy of a text file, the user's text file is copied into the work file and EDIT then operates only on the work file.
Text File	A text file is a user's external file which is copied into the work file with a TEXT command. The contents of the text file become the contents of the work file and all modifications then are performed on the work file. The work file can be saved with the present name of the text file, in which case the modifications are incorporated and the previous version of the text file is purged; or the work file can be saved under a new name and the previous version of the text file will remain unchanged.
Join File	A join file is a user's external file containing text records. It is called a join file because part or all of its content is added to the work file by way of the JOIN editor command.
Hold File	The hold file is a temporary file used for holding interim text records. It is used, for example, for temporarily holding selective portions of the work file and then adding or inserting them into other places within the work file later in the job or session.
Use File	The use file is a user's external file containing editor commands (and, optionally, text records). The content of a use file is invoked by way of the USE editor command.

Table 5-2. EDIT/3000 Standard Commands

Editor Command	Function
ADD	Enters text records into the work file from the input file and/or the hold file.
CHANGE	Changes the content of existing text records in the work file.
DELETE	Deletes characters and/or text records from the work file.
FIND	Locates a specific character position or character string in the work file.
GATHER	Moves text records from one position to another in the work file and renumbers the lines. The text records are deleted from their original positions. Also can be used to renumber all the lines in the work file.
HOLD	Copies all or part of the work file into the hold file.
INSERT	Inserts text records into the work file from the input file or the hold file at a position specified by the user.
JOIN	Copies all or part of the join file to the work file at a position specified by the user.
KEEP	Saves all or part of the work file into an MPE disc file.
LIST	Writes all or part of the work file to the output file or to any other specified file.
MODIFY	Modifies text records in the work file using one or more subcommands (delete, insert, and replace) of this command.
REPLACE	Replaces one or more text records in the work file with new text from the input file or the hold file.
SET	Alters the text editor's operating criteria.
TEXT	Copies the content of a text file into the work file, deleting the current work file content.
VERIFY	Reports the current operating criteria declared in a SET editor command (or the default criteria not declared in a SET command).
XPLAIN	Lists an explanation of all or part of the text editor command repertoire.

Table 5-3. EDIT/3000 Advanced Commands

Editor Command	Function
BEGIN	Used as the first expression in a BEGIN-END pair.
END	Terminates EDIT/3000 operation. Or, when used with a matching BEGIN command, terminates a BEGIN-END pair.
NOT	Reverses a flag after executing the command immediately following the NOT command.
OR	Sets the condition flag true, or skips the OR command and the command immediately following it if the condition flag is already true.
PROCEDURE	Calls and executes a procedure previously written by the user and stored in a segmented library (SL) file.
Q	Displays a user-defined message at the terminal.
USE	Instructs EDIT/3000 to receive commands from the use file, to send messages to the output file, and, generally, to expect text record input from the input file.
WHILE	Causes EDIT/3000 to repeat commands in a predefined command block as long as a specified condition within the text is true.
YES	Sets a flag for a WHILE command block true.
Z:= or Z:	Assigns the value of a character string variable to Z:= and uses that value whenever Z: appears as part or all of a command.

FILE COPIER

FCOPY, the HP 3000 file copier, may be used for all file copying operations on the HP 3000 Series II. With FCOPY the user can transfer files from disc, magnetic tape, or any other valid input device to disc, magnetic tape, or any other valid output device. Refer to table 5-4. FCOPY has options for translating character codes (BCD, EBCDIC, ASCII), verifying copy operations, creating new disc files, selecting file subsets by pattern matching, and for generating character, octal, and hexadecimal listings (dumps) of files. FCOPY can handle multi-file magnetic tapes; the user can skip files on a magnetic tape to get to a desired file. FCOPY works exclusively through the MPE File System.

The HP 3000 file copier is described in the *FCOPY/3000 Reference Manual* (HP part number 03000-90064).

Table 5-4. Permissable FCOPY Devices

Source	Destination
Disc File	Disc File
Magnetic Tape	Magnetic Tape
Interactive Terminal	Interactive Terminal
Punched Cards	Punched Cards
Paper Tape	Paper Tape
	Line Printer

Note: The source and destination devices listed above may be used in any combination.

SORT/MERGE

SORT, the HP 3000 sort/merge program, allows the user 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. Each record consists of a series of data fields. Sorting is based on keys (values of one or more data fields). SORT 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 user programs written in COBOL, FORTRAN, or SPL. SORT offers the following features:

- Sort of any sequential or direct-access file of any size.
- Merging of any number of sorted files.
- Ascending or descending sorts.
- Unlimited number and length of sorting keys.
- Contiguous, overlapping, or widely-separated keys.
- Operations on fixed or variable-length records of unrestricted size.
- Operations on input/output files on any devices — cards, discs, tapes, and so forth.
- Output selected from sequential records, key fields, disc addresses, or disc addresses plus key fields.

The HP 3000 sort/merge program is described in the *SORT/3000 Reference Manual* (HP part number 32214-90001).

COMPILER LIBRARY

The HP 3000 Compiler Library is a set of subroutines that provides many operations commonly needed by users programming in FORTRAN, BASIC, or SPL. These operations include:

- Extended-precision floating-point arithmetic.
- Matrix operations.
- Complex arithmetic.
- Trigonometric functions.
- Mathematical functions.
- Numeric conversions.
- Utility functions.

The Compiler Library also includes a formatter program that simplifies

input/output operations for the FORTRAN programmer. This formatter makes it unnecessary to specify precise machine operations; the programmer only specifies the format of the data, a list of variables, and a device or file.

The content of the HP 3000 Compiler Library is described in the *Compiler Library Reference Manual* (HP part number 30000-90028).

SCIENTIFIC LIBRARY

The HP 3000 Scientific Library is a set of subroutines that performs the functions most frequently required in scientific applications. It relieves the programmer of the need to create these functions for himself. The Scientific Library subroutines can be called from programs written in FORTRAN, BASIC, or SPL. They provide:

- Error and gamma functions.
- Exponential, sine-cosine, and Fresnel integrals.
- Elliptic functions and integrals.
- Bessel functions.
- Elementary statistics.
- One-way frequency distribution.
- Correlation.
- Multiple linear regression.

The content of the HP 3000 Scientific Library is described in the *Scientific Library Reference Manual* (HP part number 30000-90027).

TRACE FACILITY

TRACE, the HP 3000 trace facility, is a program debugging facility that aids the user in finding program logic errors in FORTRAN and SPL programs. TRACE helps the user follow the path of execution, the computation of values, and the manipulation of data in his programs by printing information about program identifiers (such as labels, variables, arrays, and subroutines) and structure points (critical points of passage into and out of program units) during program execution. TRACE allows the user to specify selective conditions for reporting information — for example, print data only when a variable exceeds a certain value or when it has been changed a specified number of times. The user communicates with TRACE by entering special paragraphs and sentences through his session or job input stream.

TRACE offers the following features:

- Operation in batch job or time-sharing session mode.
- Input of TRACE/3000 paragraphs and sentences in three ways:
  - Through a batch file (on cards, disc, or tape) in a job.
  - Through the terminal in a session.
  - Through a batch file (on cards, disc, or tape) in a session. This method is ideal when the user plans to issue the same directives during several successive runs of his program in session mode.
- Option to either print a report, or print a report and halt the user's program, upon satisfaction of specified conditions.

The HP 3000 trace facility is described in the *TRACE/3000 Reference Manual* (HP part number 03000-90015).

DATA ENTRY LIBRARY

The Data Entry Library (DEL/3000) is a software package that provides capabilities for designing forms, displaying forms and data on the screen of an HP 2640A or HP 2644A terminal, accepting data input from the terminal, checking the data for errors, and storing the error-free data in a disc file.

DEL/3000 includes an interactive form-maintenance program (FORMAINT) that allows the user to create forms and store them in files on disc, display, modify, or delete the forms, list all forms contained in a forms disc file, and delete an entire forms disc file.

A set of DEL/3000 procedures aids the user in writing application programs in COBOL, FORTRAN, BASIC, or SPL that display a form image and data on the terminal screen, accept data entered on the form image by a terminal operator, edit this data, and optionally store this data in a data file on disc in the format established by the user's program. The form image serves as a visible guide or template for entering the data.

Using DEL/3000, meaningful forms with "fill-in-the-blanks" format can be prepared quickly and easily to fit a wide range of applications, including:

- Order entry
- Invoice preparation
- Order status
- Sales forecasting/analysis
- Claims reporting
- Bills of lading
- Waybill generation
- Inventory receipts & reports
- Computer language coding forms (e.g. RPG)
- Material requisitions
- Bill of materials
- Labor voucher preparation
- Plant maintenance forms
- Student registration forms
- Patient history and billing
- Insurance policy applications
- Parameter lists for engineering calculations

DEL/3000 is described in detail in the *Data Entry Library Reference Manual* (HP part number 30000-90050).

DATA MANAGEMENT 6

This section discusses the data management capabilities of the HP 3000 Series II. To relate the data management systems to the overall MPE software environment, compare figure 6-1 to figure 1-1 contained in section I of this manual.

The IMAGE data management system is described in the *IMAGE Reference Manual* (HP part number 30000-90041), the QUERY data inquiry facility is described in the *QUERY Reference Manual* (HP part number 30000-90042), and the KSAM data management system is described in the *KSAM Reference Manual* (HP part number 30000-90079).

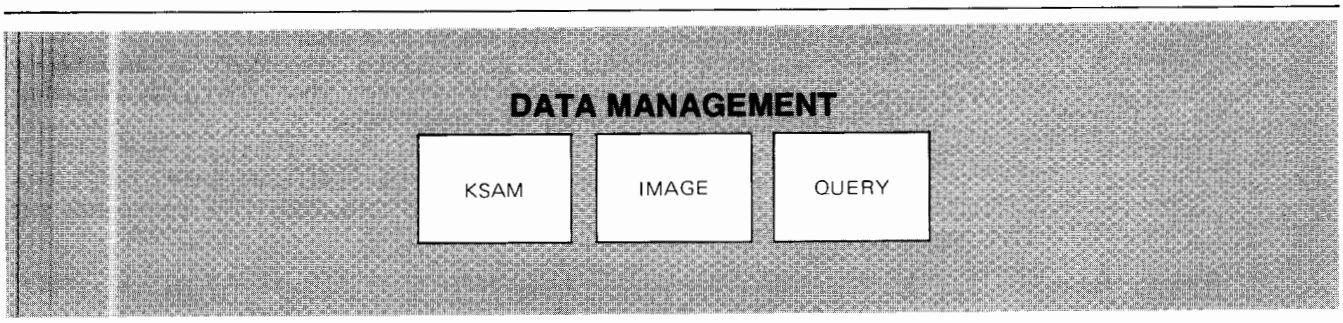


Figure 6-1. HP 3000 Series II Data Management Systems

With every passing year the sheer volume of data which must be maintained to efficiently manage a commercial, scientific, or educational enterprise grows in leaps in bounds. The development of computers as a management tool has been accompanied by a need for more sophisticated means of data management and retrieval. To provide this need a complete set of HP 3000 software products is available ranging from keyed sequential access techniques to computerized data bases and data base management.

Keyed sequential access is a method of organizing records in a file according to the content of key fields within each record.

A *computerized data base* is an organized collection of information residing on a mass-storage device attached to a computer system. Such data bases can accommodate virtually any amount of data ranging from hundreds of discrete items to literally millions. The items within a data base can be accessed sequentially (like thumbing through a drawer of index cards) or selectively through the use of search items (like using tab dividers in a drawer of index cards).

Data base management implies both a standard, but flexible, data base structure and a standard set of programs which are used for creating, accessing, and maintaining data bases. With the evolution of this concept, the programmers for each new information storage and retrieval application can now focus their efforts on the conceptual design of the data base and on writing the application programs which use it rather than on re-inventing the fundamental data base creation and maintenance programs required by the overall process.

IMAGE, the HP 3000 data base management system, provides the user with a powerful means of developing, accessing, and maintaining data bases tailored to today's corporate, industrial, scientific, and educational needs. QUERY, the HP 3000 data base inquiry facility, provides the user with a simple means of examining or altering the content of a data base or of generating complex reports interactively from a keyboard terminal.

KSAM, the HP 3000 file access method, provides the user with a means of organizing records in a file according to the content of key fields within each record.

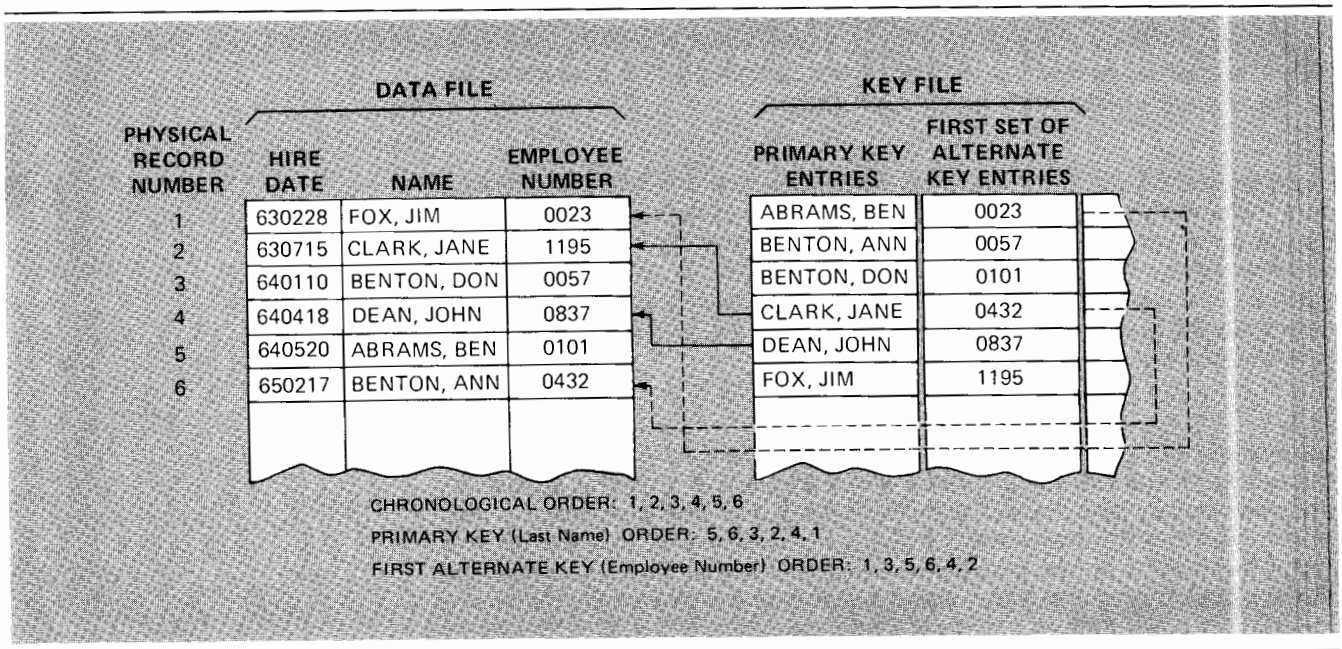
KSAM

KSAM, Keyed Sequential Access, makes it possible for the user to create and maintain disc files whose records can be accessed by the value of key fields within the data records. Each data record contains one primary key field and may include up to 15 alternate key fields. Records can be written to a KSAM file in any order without regard for key value sequences. If the records have been pre-sorted and are being written to the KSAM file in primary key order, KSAM can also check the primary key value of each record to see if it is in the proper sequence. Records can be accessed sequentially or randomly by primary or alternate key value, logical record number within a key sequence, or in chronological (physically sequential) order.

KSAM files are accessible from RPG, COBOL, FORTRAN, BASIC, or SPL programs and are compatible with the MPE STORE/RESTORE commands and FCOPY/3000.

KSAM File Structure

A KSAM file consists of two physical disc files: a data file and a key file. A KSAM data file contains all the data records. The associated key file contains one or more sets of entries that maintain the primary and alternate logical sequences of the data records. When a data record is added to a data file, a key field is inserted in the new data record. These key entries are dynamically added in ascending order. The following diagram illustrates the functional relationship between a KSAM file and its key file.



KSAM File Access

Although the KSAM files are separate, the two files that comprise a KSAM file are treated as one file by the procedures that reference the file. The data file is the only file directly referenced by a user; the key file is updated by the system to reflect any changes to the data file and is not directly accessed by the user. Therefore, from the user's point of view, accessing a KSAM file is very similar to accessing any other MPE file.

KSAM/3000 provides the following ways to store and retrieve data:

- The user can write records in logical sequence determined by primary key value or the user can write records without regard to key sequence.
- The user can read records in logical sequence determined by either the primary or an alternate key value.
- The user can read a record selected at random by the value of its primary or alternate keys.
- The user can read records in the order they were written, that is, in chronological sequence.
- The user can read a record selected by the value of its chronological record number.
- The user can update all the contents of an existing record including the contents of the primary key field.
- The user can position to a record in the file according to its key value, its chronological record number, or its record number in key sequence.

NOTE

KSAM files are stored in ascending order only, not in descending order. Character keys are ordered by the ASCII collating sequence where numbers preceded letters, not in the EBCDIC sequence where letters precede numbers. Numeric keys are ordered in algebraic order.

KSAM Components

KSAM consists of the following two logical components:

KSAM procedures and intrinsics. To assist the programmer in accessing KSAM data records, special sets of COBOL procedures, BASIC procedures, and MPE File System intrinsics (for FORTRAN and SPL) have been provided.

Note: Most of the standard MPE File System intrinsics can be used with little or no parameter changes to access and manipulate KSAM files.

KSAM utility program (KSAMUTIL). This utility program provides a set of commands designed specifically for creating and manipulating KSAM files.

BUILD — Creates a KSAM file consisting of a data file and a key file

ERASE — Clears the content of a KSAM data file and resets the pointers in the associated key file

PURGE — Deletes a KSAM file from the system

RENAME	—	Changes the name of a KSAM data file or key file
SAVE	—	Saves a session/job temporary KSAM file as a permanent file
VERIFY	—	Displays the current status of a KSAM file
HELP	—	Displays a description of all the KSAMUTIL commands
EXIT	—	Terminates KSAMUTIL and passes control back to the MPE Command Interpreter

KSAM Features

KSAM/3000 provides a number of features, including:

- Multiple Keys (up to 16 Keys)
- Duplicate Key Values
- Retrieval by Generic Key
- Retrieval by Approximate Match
- Fixed or Variable Length Data Records

MULTIPLE KEY. Each data record can contain from one to sixteen keys. Of these keys, one is required, called the primary key; and others are alternate keys. For example, in an employee record, the primary key could be the employee's social security number; alternate keys might be the employee's name, phone number, or zip code. The values in these key fields determine the order in which data records are sequenced.

Primary Key. One field in each data record is defined to contain the primary key. The value in this field determines the primary sequence of records in the data file. Records are sequenced according to this primary key unless sequencing by an alternate key or in chronological order is specifically requested.

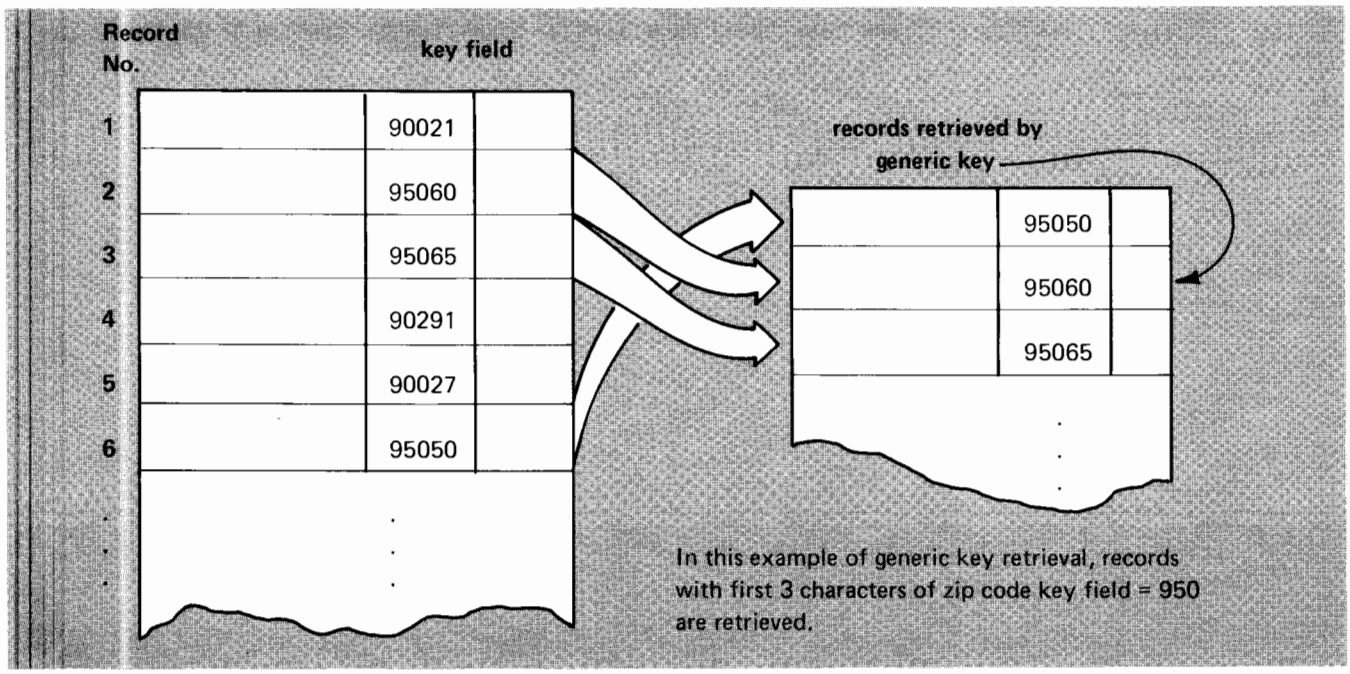
Alternate Keys. Other fields within each data record can be designated as alternate keys to be used for alternate sequencing of records. Up to 15 alternate keys can be designated for each record. The file can be sequenced in a different order for each alternate key defined for the file.

Note that alternate keys bear no hierarchial relation to each other or to the primary key. Each key is ordered in sequence by its value and type with no relation to other keys. In KSAM, sequence always means ascending sequence according to the ASCII collating sequence.

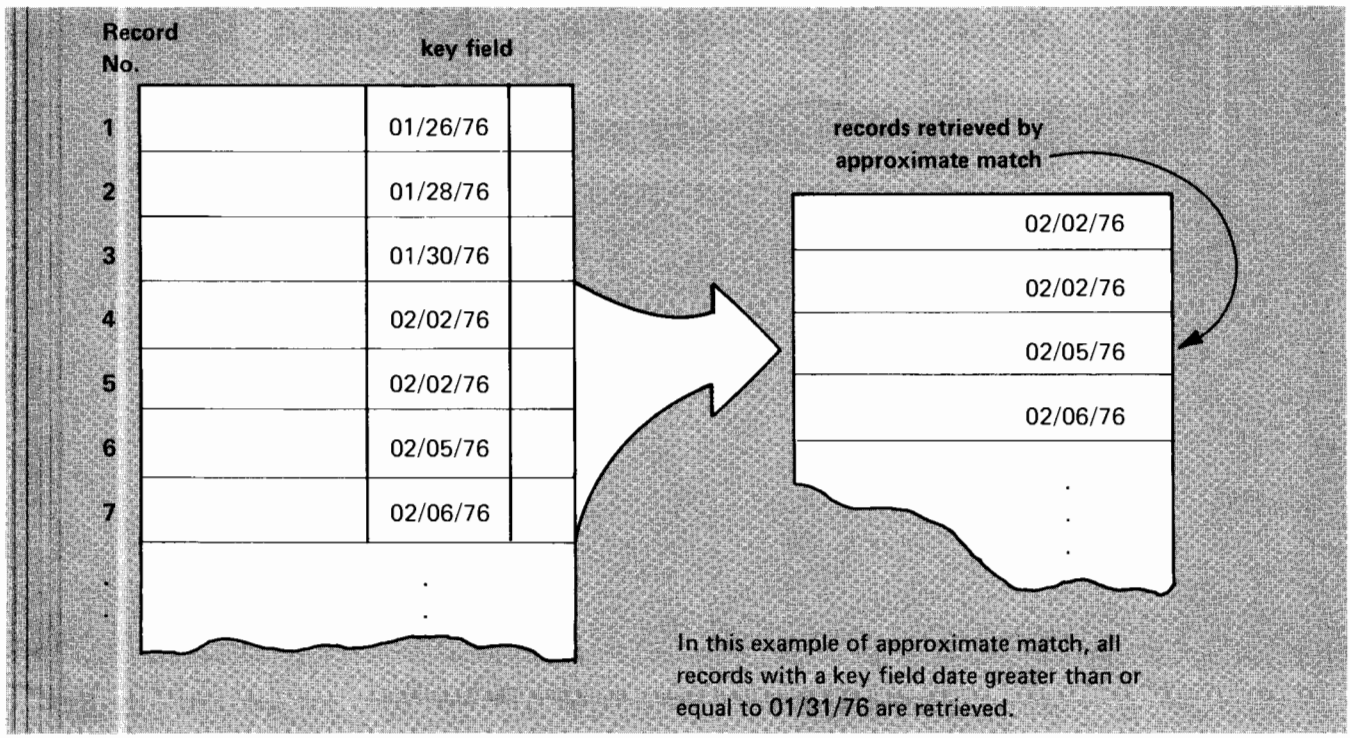
DUPLICATE KEYS. Sometimes it is essential that key values be unique (for example, a social security number), and at other times duplicate key values should be allowed (for example, a zip code). To provide for both cases, KSAM allows the user to declare that any key may have a duplicate value while disallowing duplicate key values as the default condition. Allowing or disallowing duplicate key values applies to both primary and alternate keys. Duplicates can be allowed for one or more keys while being disallowed for other keys.

GENERIC KEYS. During retrieval by key value the user can choose to use part of a key rather than the entire key. Called generic keys, such partial keys allow the user to retrieve a set of records whose key values differ in their entirety but share a common value at the beginning. Generic keys must begin at the first character of the defined key field and be shorter, not longer, than the

defined key length. Suppose a key field containing a zip code is defined as five characters long. By specifying only the first three characters for retrieval it is possible to read all records whose zip code begins with a particular group of numbers.



APPROXIMATE MATCH. When retrieving by key value, the user can specify that the key the user is looking for has a value that exactly matches a specified value, or the user can specify that it bears a certain relation to a specified value. The choices are: equal to, equal to or greater than, or greater than. The last two relations let the user search for an approximate match. For example, the user can retrieve all records with a date greater than or equal to a given date:



DATA RECORD FORMAT. Every key entry in the key file contains, in addition to the key value, a pointer to the corresponding data record in the data file. The data records can be either fixed length or variable length. If they are fixed, the data record pointer specifies a record number relative to the beginning of the file. If the records are variable length, then the pointer indicates the start of the data record as a word off-set from the beginning of the file.

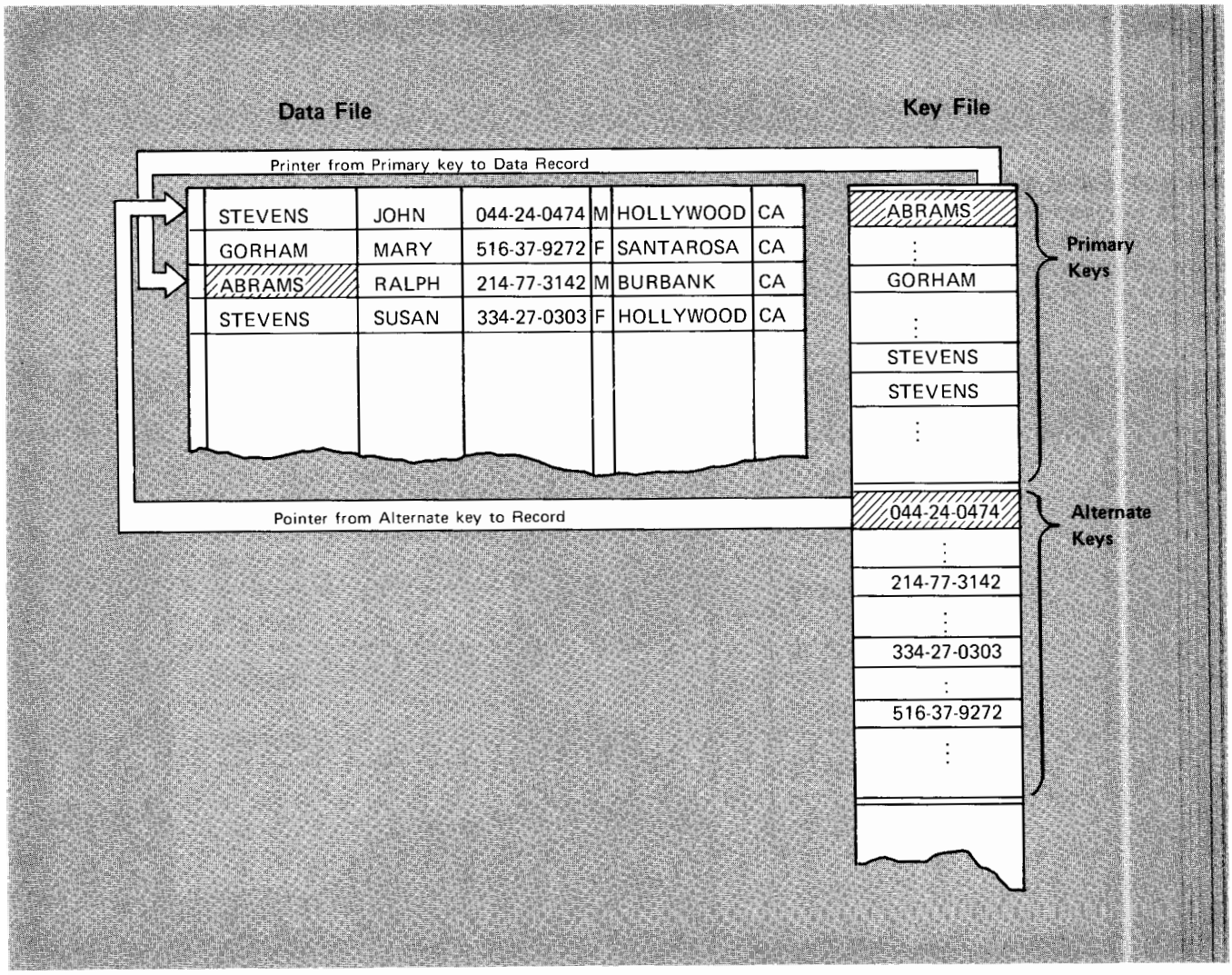


Figure 6-9. A Simplified View of the KSAM File Structure

IMAGE provides the following data base management tools:

- a data base definition language with which to write a formal description of the data base
- a schema processor which translates this description into an internal description and stores it in a disc file
- procedures which allow the user to access the data from application programs
- utility programs which maintain the data base by initializing, saving, restoring, or purging the data base.

IMAGE allows the data base designer to define a set of passwords which are used to prevent unauthorized read and write access to the various elements of the data base. It also has eight different access modes which determine the types of operations the user, and concurrent users, can perform on the data base.

IMAGE Data Base Structure

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

The following types of information can be stored as data items:

- a signed binary integer
- a COBOL signed integer (COMPUTATIONAL)
- an absolute binary number (such as SPL type logical)
- a real floating-point number (including double precision)
- an ASCII character string
- a zoned decimal number (COBOL PICTURE 9)
- a packed decimal number (COBOL COMP-3).

A *data entry* is an ordered set of related data items. The order of the data item values within a particular data entry is determined by the definition of that data entry as specified by the user when designing the data base. The length of a data entry is the combined lengths of all data items within it.

A *data set* is a collection of data entries. Each data set is referenced by a unique data set name. Figure 6-2 shows an example of three data entries in a data set named EMPLOYEE.

A *data base* is a named collection of related data sets. The data base 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. Figure 6-3 shows an example of a data base named COMPNY consisting of three data sets named LABOR, EMPLOYEE, and PROJECT.

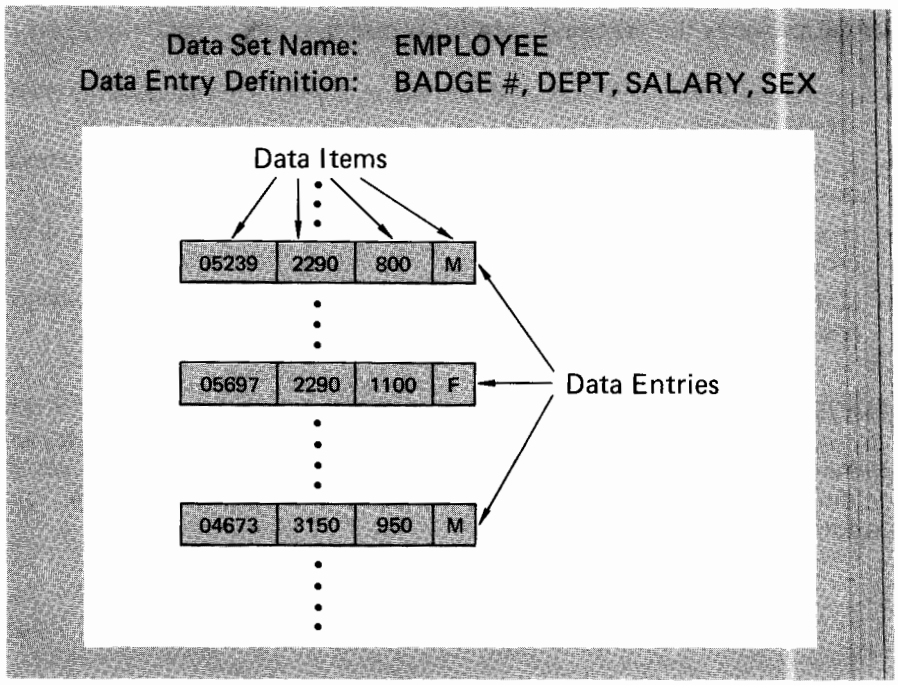


Figure 6-2. Sample Data Set

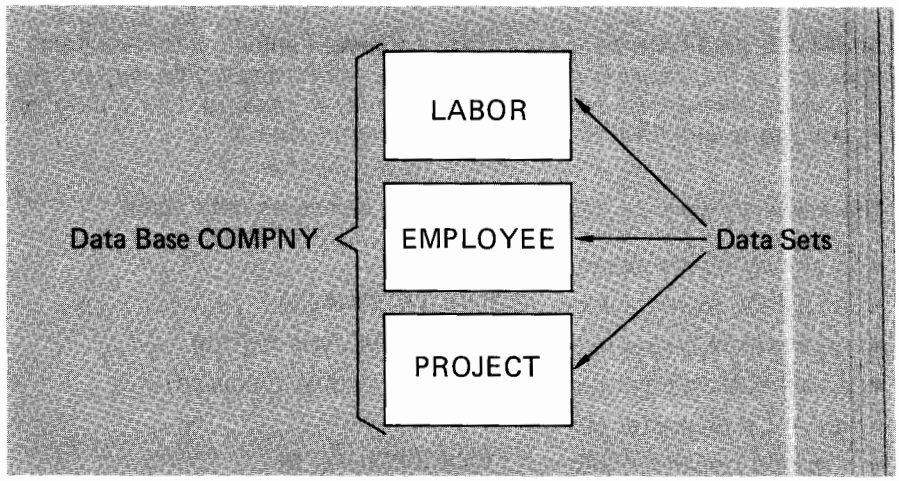


Figure 6-3. Sample Data Base

Types of Data Sets

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 an employee (such as his name, identification number, address, and salary)
- 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 sets 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 pay period for each employee in the master data set
- allow retrieval of all entries pertaining to a uniquely identifiable entity; for example, an employee identification number may be used to retrieve all data entries containing information about previous pay periods for an individual with that number

Figure 6-4 illustrates the relationship of two master data sets to one detail data set. The master data set EMPLOYEE is related to the detail data set through the search item BADGE#. The master data set PROJECT-MASTER is related to the same detail data set through the search item PROJECT.

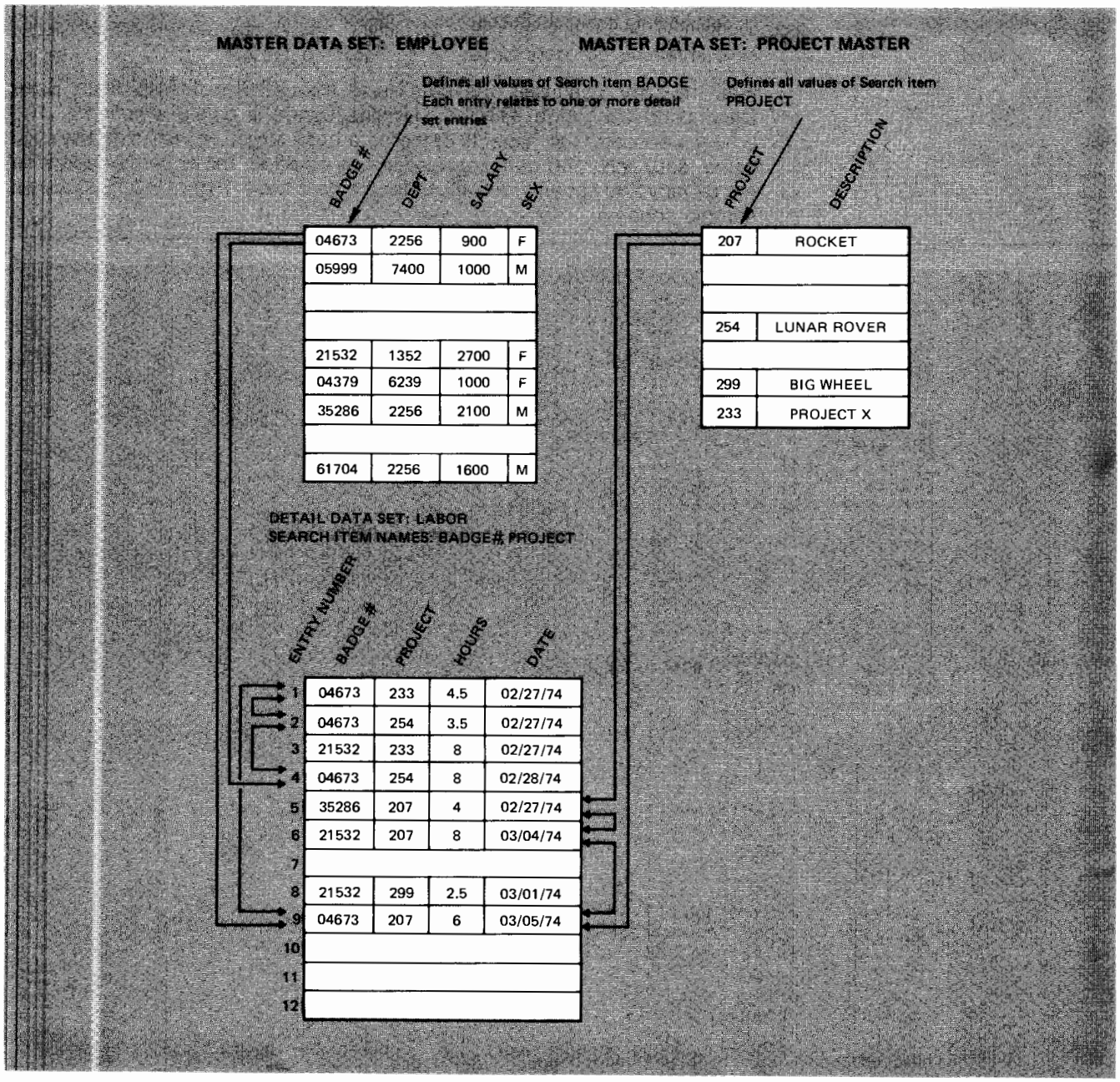


Figure 6-4. Master/Detail Data Set Relationships

IMAGE can rapidly retrieve all entries in the detail which have identical values for the search item. For example, the user can request all entries for the employee with BADGE# 04673. The arrows indicate the relationship between a master data set entry and several detail data set entries.

Each master data set may serve as an index, singly or multiply, to one or more detail data sets. Each detail data set may be indexed by one or more master data sets (or none).

Using IMAGE

The overall use of IMAGE involves five general steps:

a. Creating an external data base description.

b. Building the internal data base description.

c. Building the data base files.

d. Accessing the data base.

e. Maintaining the data base.

First the user describes the data base through the IMAGE Data Base Description Language (DBDL). This description is called a *schema*. The schema describes all aspects of the data base structure including the data items, data sets, and data set relations. In addition, the schema describes the privacy and security provisions for the data base. Figure 6-5 illustrates a sample schema.

```
BEGIN DATA BASE COMPNY;

 PASSWORDS:
  3 TRICIA;
  9 HARTHEX;

 ITEMS:
  BADGE#,   I2;
  DATE,     X8;
  DEPT,     X4;
  DESCRIP,  X16(9/9);
  HOURS,    R2;
  PROJECT,  X4;
  SALARY,   P8 ( / );
  SEX,      X2;

 SETS:

  NAME:     EMPLOYEE,MANUAL(3,9/7);
  ENTRY:    BADGE#(1),
            DEPT,
            SALARY,
            SEX;
  CAPACITY: 500;

  NAME:     PROJECT-MASTER,MANUAL(3,9/7);
  ENTRY:    PROJECT(1),
            DESCRIP;
  CAPACITY: 75;

  NAME:     LABOR,DETAIL(3,9/9,7);
  ENTRY:    BADGE#(EMPLOYEE(PROJECT)),
            PROJECT(PROJECT-MASTER),
            HOURS,
            DATE;
  CAPACITY: 10000;

END.
```

Password Definitions

Data Item Definitions

Data Entry and Data Set Definitions

Figure 6-5. Sample of IMAGE Data Base Description Language

To build an internal description of the data base, the user invokes the IMAGE Schema Processor. The Schema Processor accepts the schema as input and creates an internal representation of the schema in a disc file known as the *root file*.

The user then invokes an IMAGE utility program to build empty data files on disc, one for each defined data set.

Finally the user's application programs call IMAGE procedures to enter, retrieve, and modify data base content.

Computer Museum

Once the data base application is operational, the user creates back-up copies and performs other maintenance tasks with the IMAGE utility programs.

Accessing the Data Base

The user accesses 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 procedures locate data, maintain pointer information, manage the allocated file space, and return status information to the user. The tasks performed by the IMAGE procedures relieve the user of the "bookkeeping" normally associated with file management and allow him to concentrate on using the processing power of whatever programming language he is using. Figure 6-6 shows a sample CALL statement as it might appear in a COBOL program. Table 6-1 briefly summarizes all of the available DBMS procedures.

Some of the functions accomplished through the use of 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.

When calling an IMAGE procedure, the user must pass information about the particular data requested in the form of a parameter list; for example, the data base and data set names.

```
CALL "DBGET" USING BASE, DSET, MODE, STATUS1, LIST, BUFFER, ARGUM.
```

Figure 6-6. Sample COBOL CALL Statement

The user need not be aware of the actual physical ordering of data items within the data entry, nor even the physical location of the entry itself.

For example, the DBGET procedure retrieves an entry as it is stored in the data base and transfers to the user's buffer area only those data items requested. Figure 6-7 shows the values in the user's buffer area resulting from the read of a specific data entry with a specific data item list. Note that the resulting data item values are returned to the user's buffer area in the order requested in the procedure call regardless of the actual physical order of the values in the data entry.

Table 6-1. DBMS Procedures

Procedure Name	Function
DBOPEN	Initiates access to a data base.
DBINFO	Provides information about the data base currently being accessed (such as the name and description of a particular data item).
DBCLOSE	Terminates access to a data base.
DBFIND	Locates the first and last entries of a data chain in preparation for access to other data entries in the chain.
DBGET	Reads the data items of a specified entry.
DBUPDATE	Modifies data set entries by changing the value of data items which are not search or sort items.
DBPUT	Adds new entries to a data set.
DBDELETE	Deletes existing entries from a data set.
DBLOCK	Locks a data base temporarily to allow exclusive access.
DBUNLOCK	Unlocks a data base which was locked by a previous call to DBLOCK.
DBEXPLAIN	Prints a multi-line status message on the user's standard listing device.
DBERROR	Returns an English language status message in a buffer. The message can be printed by the user's application program.

DATA ENTRY DEFINITION: BADGE #, DEPT, SALARY, SEX

DATA ENTRY:

09095	1093	3300	M

DBGET PROCEDURE DATA ITEM LIST: BADGE #, SEX, DEPT

DATA ITEM VALUES RETURNED IN USER'S BUFFER:

09095	M	1093

Figure 6-7. Values Returned to User's Buffer

At any given time a record may or may not contain an entry. IMAGE maintains internal information indicating which records of a data set contain entries and which do not.

Each time a request for retrieval is made, the IMAGE procedure must determine the data entry from which to extract the desired data items. The

determination is based upon the type of access requested in the IMAGE procedure call. There are four modes of access available:

- *Serial* In this mode IMAGE starts at the most recently accessed record and sequentially examines successive records until the closest data entry, if any, is found. Both forward and backward serial access are 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 is retrieval of master data set entries based upon the search item (key) value. It is not necessary for the user to know where the desired entry is located within the data set. IMAGE uses the search item value as input to an address formula to calculate the record number of the desired entry.
- *Chained* This type of access consists of successive retrieval of all entries in a detail data set which contain the same search item value.

The example in figure 6-4 can be used to illustrate chained access. All pieces of work done by the employee whose badge number is 04673, or all the work done on project 207, can be rapidly retrieved using chained access. By contrast, a program using a non-search item such as DATE to locate entries containing specific DATE values must serially search the data set. The selection of data items as search items depends upon expected usage, frequency of retrieval, number of distinct values, and other criteria.

Maintaining the Data Base

The user initializes and maintains a data base by executing IMAGE Data Base Utility Subsystem (DBUS) programs. These programs help maintain data base integrity.

IMAGE utility programs also perform the following functions:

- Copy the data base from disc files to magnetic tape, thus, providing back-up copies of the data base.
- Reload a data base from magnetic tape back into the data base disc files.
- Erase data base content while maintaining the root file and the data files of the data base.
- Purge the entire data base including its root file and data files.

Table 6-2 briefly summarizes all of the available DBUS programs.

The user can copy either the data base content or the data base content and related structural information to magnetic tape. The first option allows the user to store the content of a data base on magnetic tape and then restore the data *into a new data base structure*. To do this, the user first stores the content of a data base on magnetic tape using the DBUNLOAD program, then redefines the data base structure (using DBDL and the Schema Processor), and finally restores the data from magnetic tape into the new data base structure using the DBLOAD program.

Table 6-2. DBUS Programs

Program Name	Function
DBUTIL	Allocates and initializes all data sets of a data base, or reinitializes all data sets of a data base, or purges the root file and all data sets of a data base.
DBSTORE	Stores the root file and all data sets of a data base on magnetic tape in a format compatible with the MPE :STORE and :SYSDUMP commands.
DBRESTOR	Restores the root file and all data sets from a magnetic tape created by the DBSTORE program (or by the MPE :STORE or :SYSDUMP commands) to disc.
DBUNLOAD	Dumps all the data entries of each data set of a data base to a specially formatted magnetic tape. This logical dump of the data base is useful for changing the data base structure.
DBLOAD	Loads an already created data base with the content of a DBUNLOAD magnetic tape.

QUERY

QUERY, the HP 3000 data base inquiry facility, provides a simple method of accessing an IMAGE/3000 data base without programming effort. QUERY allows users to do the following:

- add data
- modify or delete data values on-line
- retrieve data which meets selection criteria
- report on the data retrieved

To perform these operations, the user enters simple commands consisting of English-language key words such as FIND and REPORT.

The user only needs to know the relationships of the data base elements and not the structure of the disc files; QUERY finds the data and performs the operations in response to commands using the data set and data item names specified.

QUERY can be used from either a terminal or a batch input device such as a card reader. If QUERY is run in an MPE session at a terminal, it operates interactively, prompting the user for commands, issuing error messages when an error occurs, and printing other information of interest to the user. To operate QUERY in batch mode (as an MPE job) the user enters one command per record from a batch input device. If an error occurs, QUERY ignores the command that is in error.

QUERY output can be directed to any appropriate device by using a combination of QUERY commands and simple MPE :FILE commands. For example, output from the QUERY REPORT command can be directed to a line printer, magnetic tape, disc, or card punch.

Applications

QUERY can be used for a variety of applications, complex or simple. Some of these applications are:

- *Casual inquiry of the data base.*

  Because QUERY requires no programming effort, the user can search a data base for information without writing a program or waiting for a periodic batch run.

- *On-line data updates.*

  With QUERY the user can enter, delete, and modify data on-line, directly from the terminal to the IMAGE data files. No intermediate steps need be taken. The data base is updated quickly; the user need not wait for periodic update runs to enter the latest information into his data base.

- *Report generation.*

  QUERY's powerful report formatting capability enables the user to build reports with header and column labels, page numbers, and group labels. In addition, the user can sort entries through multiple fields, as well as total, average, or count columns of numeric data values. Arithmetic calculations can be performed using the ten QUERY registers and the results printed in the report.

- *Application program debugging.*

  If the user is developing programs which access IMAGE data bases through the IMAGE procedures, QUERY makes an excellent debugging aid. The user can manipulate data in the data base using his program and then use QUERY to access data in the data base to test the results.

Features

In addition to the features mentioned above, QUERY offers the following:

- *Selection of data through logical comparisons.*

  QUERY locates entries based upon logical criteria specified in the FIND command. Many conditions can be specified in one command. For example, the command:

  ```
  FIND COMMISSION GT "2000" AND OCCUPATION IS "SALESMAN" END
  ```

  instructs QUERY to locate those data entries with the value of data item COMMISSION greater than 2000 and OCCUPATION equal to SALESMAN. Only those entries which fulfill the two criteria are located.

- *Creation and storage of procedures.*

  A frequently used or complex command can be stored as an individual procedure in a command file (known as the Proc-file) and then called by name. Each command is stored as a separate procedure. The procedure name can then be used in place of the command parameters, thus, removing the need to re-enter frequently used or lengthy commands.

- *Display of the data base structure.*

  The data base structure can be displayed through the use of the FORM command. QUERY prints out the data items and data sets you may access and shows the relationship between them. Only those entities to which you have at least read access are displayed.

- *Storing Commands in a file.*

  A sequence of commands can be stored in a file and executed at any

time by entering a single command, XEQ, as part of a session or a job.

- *Security provisions.*

  QUERY adheres to all of the security provisions included in the IMAGE data base by the data base designer. After QUERY is invoked, the user must enter a designer-supplied password. This word determines which data entries and data items the user may access in the data base.

Command Set

The QUERY command set consists of 23 commands. Table 6-3 shows each command and summarizes what it does.

Table 6-3. QUERY Command Set

Category	Commands	Function
Environment	DATA-BASE=	Specifies the data base to be accessed.
	DATA-SETS=	Specifies the data sets to be accessed.
	PROC-FILE=	Specifies which Proc-file to be used.
	OUTPUT=	Specifies the output device to be used for command output.
	DEFINE	Indicates the current state of the Environment commands and prompts you for changes.
	PASSWORD=	Specifies access password.
	MODE=	Specifies access mode.
Utility	FORM	Displays the structure of the data base being accessed.
	HELP	Lists a description of the QUERY command set.
	EXIT	Terminates QUERY execution.
	VERSION	Displays version, update, and fix information.
Location	FIND	Locates data entries in the data base according to your specifications.
Updating	ADD or UPDATE ADD	Adds data entries to the data base.
	DELETE or UPDATE DELETE	Removes data entries from the data base.
	REPLACE or UPDATE REPLACE	Modifies data items in data entries.
Reporting	REPORT	Reports about the entries located by the FIND command.
	LIST	Prints entries with automatic formatting.
Procedure	CREATE	Stores procedures in the Proc-file.
	DISPLAY	Displays procedures stored in the Proc-file.
	ALTER	Modifies procedures currently stored in the Proc-file.
	DESTROY	Removes procedures from the Proc-file.
	RENAME	Changes procedure name.
Operation	XEQ	Executes QUERY commands from a file.

Using QUERY

Figure 6-8 shows a sample QUERY session. In this example, the data base is named STORE. It consists of several data sets including one named PRODUCT and one named CUSTOMER. PRODUCT contains the stock number and description of each product the store sells. CUSTOMER contains information about the store's customers, such as their account number, name, address, and credit rating.

The numbered paragraphs below annotate the session depicted in figure 6-8. User input is distinguished from computer output with underlines.

1 LOGGING ON TO MPE

This command initiates the user session. MPE responds to a correct log-on procedure by printing a heading.

2 RUNNING QUERY/3000

The :RUN command starts QUERY execution. QUERY first issues a prompt for a command. The DEFINE command is used to specify the data base to be accessed, the password, mode of access, and so forth.

3 INVESTIGATING THE DATA BASE STRUCTURE

The FORM command requests information about the data base structure. In this example, information about the PRODUCT data set is requested. The information printed includes the data set name and type, names of the data items in the PRODUCT data set, and the capacity and number of entries in the data set.

4 ADDING ENTRIES TO A DATA SET

The ADD PRODUCT command adds data entries to the data set named PRODUCT. A prompt is issued for the value of each data item in the entry by printing the name of the data item. The value for each item is entered following the prompt.

5 LISTING DATA ITEM VALUES

The LIST command can be used to list the value of each data item in all entries of the data set or to list entries selectively based on the values of data items. In the example, entries containing STOCK# values less than 000050000 are listed.

6 LOCATING DATA ENTRIES

The FIND command locates data entries. In this example, all entries containing the DESCRIPTION data item are located. The FIND command also selectively locates entries if particular values of data items are requested. (See 8 .)

7 PRINTING A FORMATTED REPORT

Once the desired data entries have been located using the FIND command, the REPORT command prints the data entries in the report format specified. In the example, the values are sorted by STOCK# and printed.

8 REPLACING DATA ITEM VALUES

The entry to be changed is located with the FIND command. The REPORT ALL command prints the value of each item in the entry.

The NOPAGE parameter suppresses pagination. The REPLACE command is used to change the value of an item in the entry. In the example, the user first types the HELP command to request the format of the REPLACE command. Then the user replaces the INITIAL and CITY data item values. Another form of the REPORT ALL command is used to check the new values in the entry. By entering a character after "ALL", the user suppresses the data item names.

9 TERMINATING EXECUTION

The EXIT command terminates QUERY execution and returns control to the operating system. The operating system prompts for a command by printing a colon (:).

10 LOGGING OFF

The MPE BYE command logs the user off the system. MPE prints the amount of CPU time used, the number of minutes the user was connected to the system, and the current date and time.

```
:HELLO BROWN.DATAMGT                                   1
 SESSION NUMBER = #S12
 THU, MAR 18, 1976,  8:18 AM
 HP32002A.00.00

:RUN QUERY.PUB.SYS                                     2

 HP32216A.03.00 QUERY/3000  THU, MAR 18, 1976,  8:19 AM


 QUERY/3000 READY

>DEFINE
DATA-BASE = >>STORE
PASSWORD = >>DO-ALL
MODE = >>1
DATA-SETS = >>
PROC-FILE = >>PROCX
OUTPUT = TERM
OUTPUT = >>
>FORM PRODUCT                                          3


DATA BASE: STORE                          THU, MAR 18, 1976,  8:20 AM


SET NAME:
   PRODUCT,MANUAL

      ITEMS:
         STOCK#,                U8              <<KEY ITEM>>
         DESCRIPTION,           X20

      CAPACITY: 5003             ENTRIES: 3
```

Figure 6-8. Sample QUERY Session (Sheet 1 of 3)

```
>ADD PRODUCT                                                    4
STOCK#            =>>00001234
DESCRIPTION       =>>OAK BARREL


STOCK#            =>>00001235
DESCRIPTION       =>>METAL BOX


STOCK#            =>>//
>LIST PRODUCT                                                   5


STOCK#     DESCRIPTION

00023400   PAPER BAG
00001235   METAL BOX
34000000   WOODEN CRATE
00001234   OAK BARREL
99310000   CARDBOARD CARTON


>LIST PRODUCT FOR STOCK# ILT 00005000


STOCK#     DESCRIPTION

00001235   METAL BOX
00001234   OAK BARREL


>FIND ALL DESCRIPTION                                           6
USING SERIAL READ
5   ENTRIES QUALIFIED
>REPORT                                                         7
>>H1,"LIST OF ALL PRODUCTS",30,SPACE A2
>>H2,"STOCK NUMBER",14,SPACE A1
>>H2,"PRODUCT DESCRIPTION",40
>>D1,STOCK#,12
>>D1,DESCRIPTION,46
>>S,STOCK#
>>END

            LIST OF ALL PRODUCTS
   STOCK NUMBER        PRODUCT DESCRIPTION

      00001234                 OAK BARREL
      00001235                 METAL BOX
      00023400                 PAPER BAG
      34000000                 WOODEN CRATE
      99310000                 CARDBOARD CARTON
```

Figure 6-8. Sample QUERY Session (Sheet 2 of 3)

```
>FIND ACCOUNT IS 34567890
ACCOUNT            IS A MEMBER OF THESE SETS:
CUSTOMER,SALES
WHICH SET DO YOU WISH TO USE?
>>CUSTOMER
1  ENTRIES QUALIFIED
>REPORT NOPAGE ALL
ACCOUNT            =34567890
LAST-NAME          =NEWTON
FIRST-NAME         =NORBERT
INITIAL            =M
STREET-ADDRESS     =13 BARRINGTON LANE
CITY               =CARMLE
STATE              =CA
ZIP                =93921
CREDIT-RATING      = 4.00000
>HELP REPLACE FORMAT

FORMAT -
   REPLACE,DATA ITEM NAME = "VALUE";
            [DATA ITEM NAME = "VALUE";]
            .
            .
            .
            [DATA ITEM NAME = "VALUE";]
            END

>REPLACE,INITIAL="M.";
>>CITY="CARMEL";
>>END
>REPORT ALL,Y

34567890
NEWTON
NORBERT
M.
13 BARRINGTON LANE
CARMEL
CA
93921
 4.00000

>EXIT

 END OF PROGRAM
:BYE

CPU (SEC) = 35
CONNECT (MIN) = 12
THU, MAR 18, 1976,  8:30 AM
END OF SESSION
```

Figure 6-8. Sample QUERY Session (Sheet 3 of 3)

DATA COMMUNICATIONS 7

This section summarizes the data communications capabilities of the HP 3000 Series II. To relate the data communications programs to the overall MPE software environment, compare figure 7-1 to figure 1-1 contained in section I of this manual.

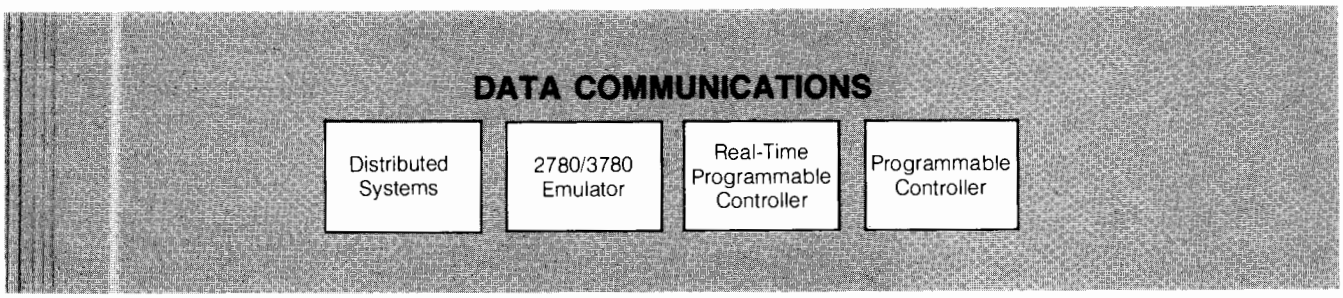


Figure 7-1. HP 3000 Series II Data Communications Programs

DISTRIBUTED SYSTEMS/3000

DS/3000 is a set of software enhancements to MPE (and hardware interfaces) which provide direct communications between several HP 3000 Series II Computer Systems. The connection can be made over hardwired (coaxial cable) lines or over public telephone lines.

DS/3000 was designed to be both easy to use and versatile enough to adapt to a broad spectrum of commercial/industrial applications. Most of the powerful DS/3000 capabilities are available to the user with no programming effort as soon as the network hardware and software are installed.

DS/3000 Capabilities

The power and flexibility of DS/3000 as well as its ease of use are demonstrated in the following DS/3000 capabilities:

Remote Command Processing

With remote command processing the user has the ability to execute a command in a remote HP 3000 while connected to a local HP 3000. The entire set of MPE commands can be executed remotely. To execute a command in the remote HP 3000 the user can use the following construct:

:REMOTE command

where *command* is the desired MPE command in its normal format. For example, if the user wishes to display the names of all disc files residing in the remote log-on group and account in the remote HP 3000, the user would issue a remote LISTF command as follows:

:REMOTE LISTF

FILENAME

DATA1 DATA5 DATA6 FILE3 SOURCE1

Remote File Access

With DS/3000 the user can use the full capabilities of the standard file system for sharing both local and remote files. This is done by simply specifying the device parameter of the desired system. For example, assume that there is a COBOL source file named SOURCE1 residing on a disc connected to a remote HP 3000 Series II computer system and the user wishes to compile, prepare and execute that program on a local HP 3000 system. The user can locally perform the compilation, preparation, and execution of the remote source file with the following command sequence:

```
:HELLO USER.ACCOUNT                    (logs onto local HP 3000)
 SESSION NUMBER =  #598
 TUE, FEB 18, 1977, 12:51 PM
HP32002A.00.A1

:REMOTE HELLO RUSER.RACCOUNT;DSLINE=CPUC

DS LINE NUMBER = 3                     (logs onto remote HP 3000)

 SESSION NUMBER = #S17
 TUE, FEB 18, 1977, 12:52 PM
 HP32002A.00.A1

:FILE SOURCE;DEV=CPUC#DISC             (defines file SOURCE1 as
:COBOLGO *SOURCE1                       being on remote disc file)
```

Program-to-Program Communications (PTOP)

The DS/3000 program-to-program communications capability gives a programmer the ability to write application programs which run simultaneously in separate computers that communicate efficiently with one another. This program-to-program capability is a set of nine new intrinsics that makes it possible for two or more user programs residing in separate HP 3000s to exchange data and control information directly and efficiently with one another over DS/3000 communications links.

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) is always responding to requests received from the master. Those intrinsics used within a master program are summarized in table 1 and those used in a slave program are summarized in table 2.

This master/slave relationship is truly dynamic in that it is determined solely by the design of the user's application programs. Each HP 3000 in a DS/3000 network can be used alternately as master and slave; if need be, a particular HP 3000 can even be used as a master and slave simultaneously.

Table 1. DS/3000 Master Program-to-Program Intrinsics

Intrinsic Name	Function
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.
PWRITE	Sends a block of data to the remote slave program.
PCONTROL	Transmits a tag field (containing user-defined control information) to the remote slave program and receives a tag field back from the slave.
PCLOSE	Terminates (kills) the remote slave process.
PCHECK	Returns an integer code specifying the completion status of the most recently executed master program-to-program intrinsic.

Computer Museum

Table 2. DS/3000 Slave Program-to-Program Intrinsics

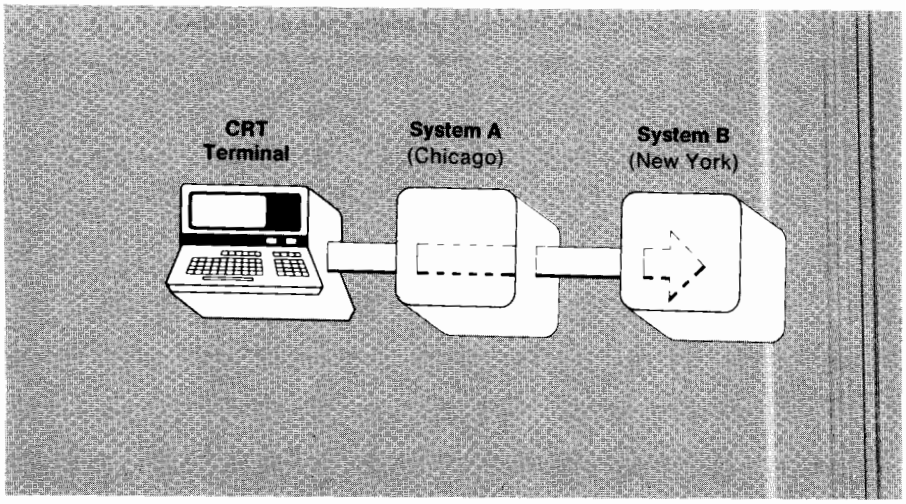
Intrinsic Name	Function
GET	Receives the next request from the remote master program.
ACCEPT	Accepts (and completes) the request received by the preceding GET intrinsic call.
REJECT	Rejects the request received by the preceding GET intrinsic call.
PCHECK	Returns an integer code specifying the completion status of the most recently executed slave program-to-program intrinsic.

The use of these three capabilities results in the following features:

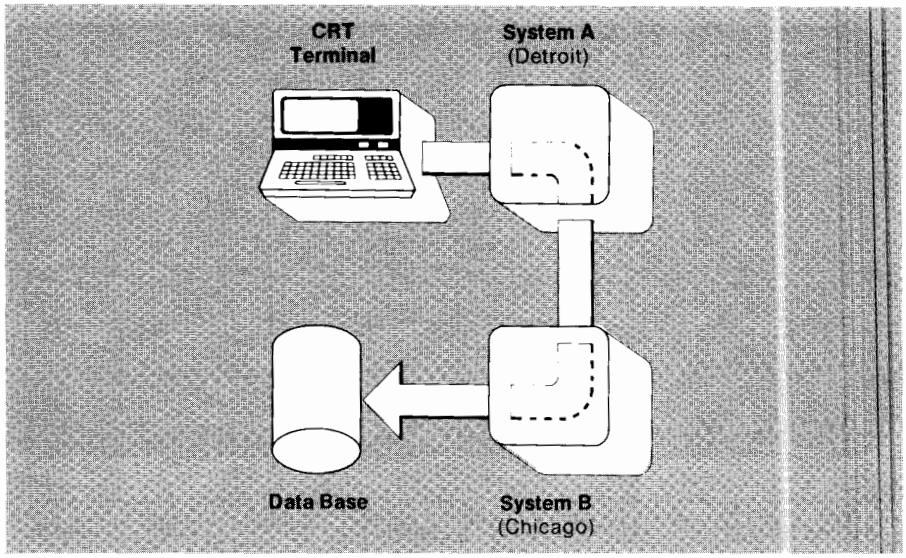
Virtual terminal capability. This feature allows terminals physically connected to one system to operate logically as though they were connected to another. When the user is seated at a terminal in a DS/3000 network, the user literally has the hardware and software resources of the entire network at his fingertips.

Virtual terminal capability is available to all terminals in a DS/3000 network and can be used simultaneously in either direction over any available communications lines in the network.

Simultaneous local and remote processing: Within each system in a DS/3000 network a wide variety of processing can be in progress simultaneously, including local and remote batch, local and remote transaction processing, local and remote interactive problem solving, 2780/3780 remote job entry, and inter-system program-to-program communication.



Remote Data Base access: The user can remotely use the Hewlett-Packard QUERY/3000 facility to interactively store, modify, retrieve, or report on data in IMAGE/3000 data bases that reside on remote HP 3000 systems. With the DS/3000 program-to-program communications capability the user can run programs on one system that indirectly access and update an IMAGE data base residing on another system.



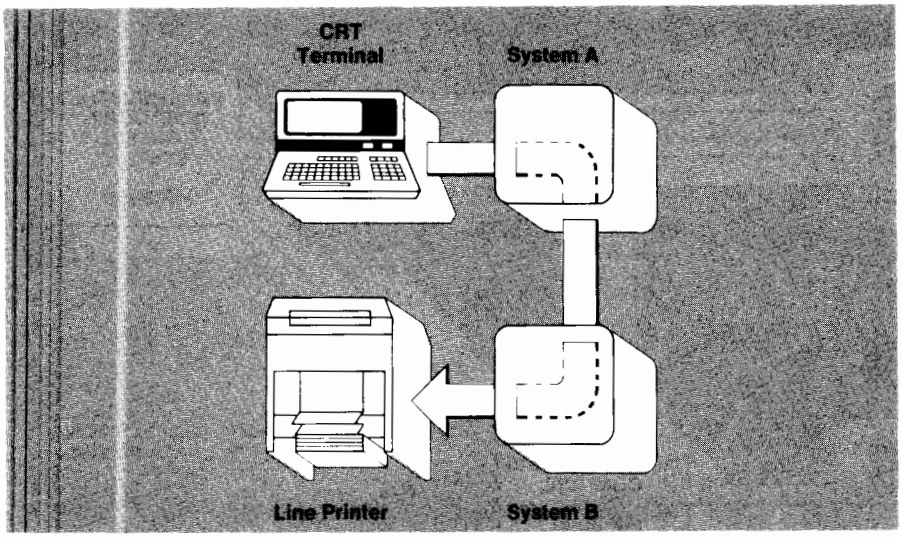
Inter-system data transfer: Within a DS/3000 network, the HP 3000 File Copier (FCOPY/3000) can be used to copy files from one HP 3000 system to another. The remote file access capability of DS/3000 makes it easy for programs running in one system to access the files and devices of any other system in the network. The DS/3000 program-to-program communications capability can be used for performing high-speed inter-system block data transfers.

Bidirectional interleaving: Each physical communications line in a DS/3000 network can be used concurrently for many different batch and interactive applications originating from either end of the line. DS/3000 automatically multiplexes and demultiplexes all data being transferred over the line. In interleaved usage, each application is essentially unaware that anyone else is using the same line.

Choice of switched, leased, and hardwired lines: DS/3000 offers a choice of switched, leased, or hardwired lines and allows the user to mix these line types freely throughout his network. The user can choose the most economical type of line for each particular link in the network. For further information refer to the HP 3000 Configuration Guide for synchronous modems.

High data throughput: Using a hardwired line (coaxial cable), line speeds up to 2.5 million bits per second (up to 1000 feet) can be achieved. Using common carrier facilities (switched or leased lines), data can be transferred at a rate of up to 9600 bits per second depending upon line conditioning and choice of modem. When a particular application requires the maximum possible data throughput over a line, DS/3000 allows the user to obtain exclusive access to the line. In that case no one else can obtain access to that line until the exclusive user releases it. Exclusive access is more efficient than non-exclusive access; however, the effective data transfer rate is a function of the application.

Peripheral sharing: With DS/3000 the user can share the use of expensive peripherals among several systems via communications lines, thus achieving a significant cost savings without sacrificing capability.



Network design flexibility: DS/3000 has been designed so that the master/slave relationships between systems in the network are truly dynamic. Systems can alternate freely from master to slave status or operate in both modes simultaneously.

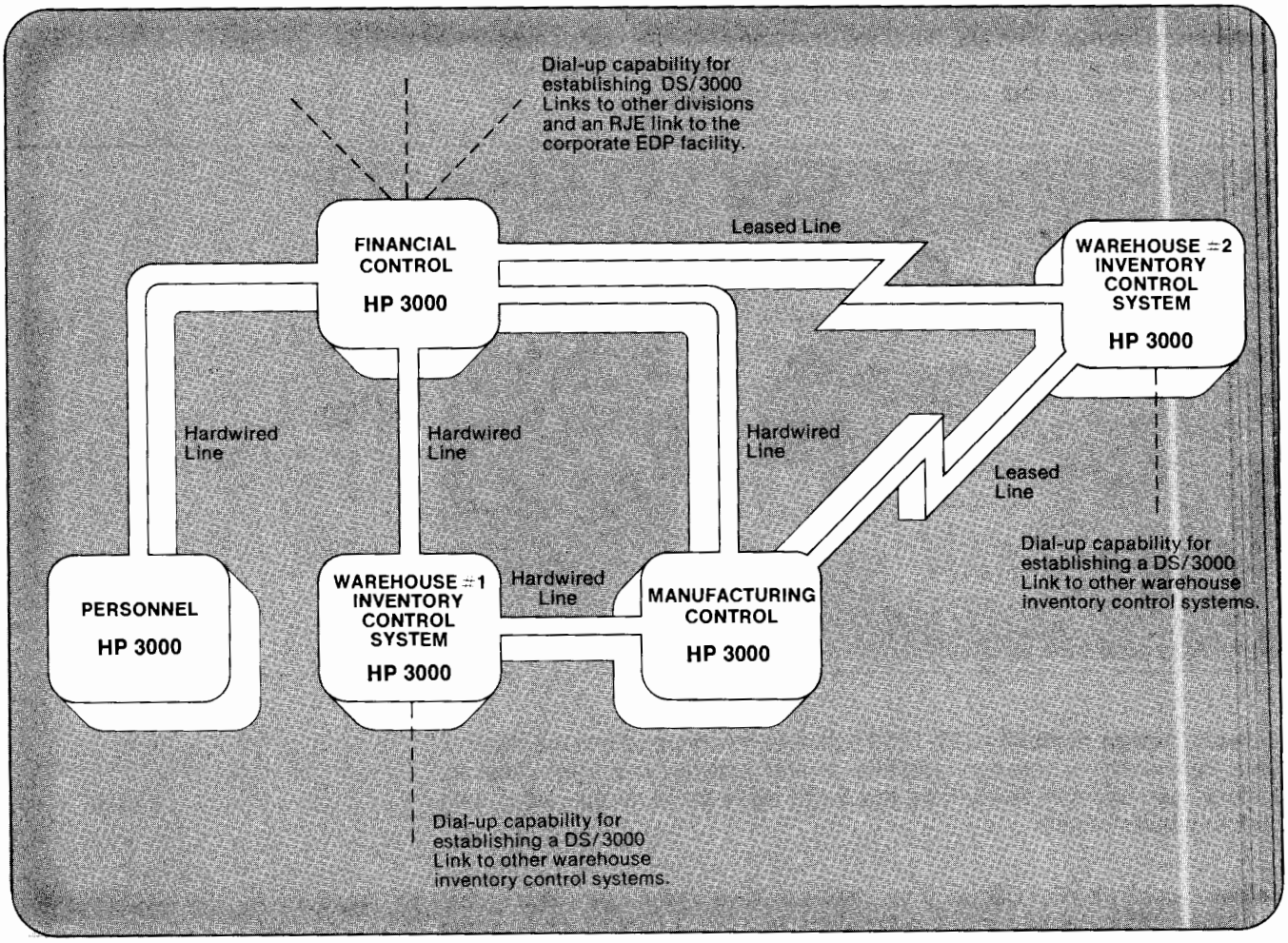
Network control: With DS/3000 the user can have a computer network that distributes computer power to the locations where it is needed while, at the same time, maintains control at a central location, if desired. The remote command processing feature allows a single system manager to control all satellite HP 3000 systems from the central EDP facility.

Network security: Within a DS/3000 network the accounting structure and file security features of MPE II provide total protection against unauthorized use of HP 3000 systems and their disc files. To initiate a remote session, the user must supply user and account names that are considered valid by the accounting structure of the remote HP 3000. Passwords can be attached to user names, account names, group names, and disc file names to provide a multi-level security scheme within each HP 3000 in the network. In addition,

the MPE II File System can be used to attach access levels to individual disc files, thus strictly limiting read, write, or read/write access to a particular subset of users.

Network reliability: DS/3000 uses cyclic redundancy check (CRC) characters to determine whether each particular block of data is transferred successfully from one system to another. If this CRC character indicates that an error occurred, the block is automatically retransmitted. If an error persists through the maximum number of retransmissions, DS/3000 considers the error to be irrecoverable and reports it to the user.

In addition, DS/3000 contains a diagnostic program that the customer can execute to determine if his network configuration is working properly.



A Typical DS/3000 Network

2780/3780 EMULATOR

The HP 2780/3780 Emulator lets the user transfer data between an HP 3000 Series II and a variety of remote processors in a full multiprogramming environment over the public telephone (switched) network or a private leased line. The emulator can transfer data at rates of up to 4800 bits-per-second, depending upon the choice of MODEM. However, such determination must be made on an individual basis.

The emulator, which runs under the control of MPE, makes the HP 3000 Series II appear to the remote processor as either an IBM 2780 or 3780 Data Transmission Terminal. The emulator is more flexible than the IBM terminals in that it allows the user to use a greater variety of input/output devices, including disc and magnetic tape.

The remote processor can be any of the following:

- any computer which supports the IBM 2780 or 3780 Data Transmission Terminals
- an IBM 2780 or 3780
- another HP 3000 Series II which is also using an HP 2780/3780 Emulator

The user can operate the emulator in either the job (batch) or session (interactive) mode. Ten emulator commands allow the user to control the sequence of input/output processing. These commands are summarized in table 7-1.

Provided that the HP 3000 Series II has more than one Synchronous Single Line Controller, several people may use the emulator concurrently. The number of concurrent users is limited by the number of Synchronous Single Line Controllers which are available. Prior to invoking the emulator, the user specifies which Synchronous Single Line Controller he wishes to use by issuing an MPE :FILE command.

Table 7-1. HP 2780/3780 Emulator Commands

Command	Function
#RJLINE	Defines the characteristics of the communications line.
#RJIN	Transmits input data from the HP 3000 to the remote processor.
#RJIO	Transmits a one-line message from the HP 3000 to the remote processor and accepts a variable-length message in return.
#RJOUT	Requests and processes routed output from the remote processor.
#RJLIST	Requests and processes unrouted list output from the remote processor.
#RJPUNCH	Requests and processes unrouted punched output from the remote processor.
#RJINFO	Displays information about the communications line that may be helpful in diagnosing problems.
#RJDEBUG	Invokes the MPE Debug facility.
#RJEOD	Transmits an EOT control character (end-of-file) to the remote processor.
#RJEND	Terminates the HP 2780/3780 Emulator job or session.

Note: The terms "routed" and "unrouted" refer to the presence or absence of component select codes in conjunction with output transmitted from the remote processor. This concept is discussed in detail under the descriptions of the #RJOUT, #RJLIST, and #RJPUNCH commands in the emulator reference manual.

IBM 2780 Data Transmission Terminal

The IBM 2780 Data Transmission Terminal was IBM's original low-cost remote batch processing terminal and was the first data processor to use IBM Binary Synchronous Communications telecommunications line protocol (commonly referred to as BSC or bisynch). Physically, the IBM 2780 consists of the following hardware elements:

- A control unit, a bisynch adapter, and buffers.
- A card reader capable of reading up to 400 cards-per-minute.
- A card punch (optional) capable of punching up to 355 cards-per-minute.
- A line printer (optional) capable of printing up to 300 lines-per-minute.

Functionally, an IBM 2780 is capable of the following:

- Transmitting or receiving in ASCII, EBCDIC, or six-bit transcode.
- Accepting horizontal tabulation codes, thereby making it possible to process line printer output more efficiently (this capability is optional on the IBM 2780).

- Accepting vertical forms control codes, thereby facilitating the formatting of line printer output.
- Transmitting or receiving data in the transparent mode, thereby making it possible to transmit or receive binary data (this capability is optional on the IBM 2780 and applies only to the EBCDIC mode).
- Performing short-record truncation. This makes it possible to suppress trailing blanks from card input, thereby increasing the transmission efficiency. However, the user must indicate the start of each trailing blank field by including an EM control character in the data.
- Transmitting or receiving blocked data, thereby increasing the transmission efficiency. The maximum number of records allowed per block is 2 (as an option, this maximum can be raised to 7).

IBM 3780 Data Communication Terminal

Like the IBM 2780, and IBM 3780 Data Communication Terminal is a control unit with a set of input/output devices. Physically, the IBM 3780 consists of the following hardware elements:

- A control unit, a bisynch adapter, and buffers.
- A card reader capable of reading up to 600 cards-per-minute.
- A card punch (optional) capable of punching up to 359 cards-per-minute.
- A line printer capable of printing up to 425 lines-per-minute.

Functionally, the IBM 3780 is capable of the following:

- Transmitting or receiving in ASCII or EBCDIC.
- Accepting horizontal tabulation codes, thereby making it possible to process line printer output more efficiently.
- Accepting vertical forms control codes, thereby facilitating the formatting of line printer output.
- Transmitting or receiving data in the transparent mode, thereby making it possible to transmit or receive binary data (this capability is optional on the IBM 3780 and applies only to the EBCDIC mode).
- Performing short-record truncation. This makes it possible to suppress trailing blanks from card input, thereby increasing the transmission efficiency. This function is performed automatically; the user does not need to include EM or IRS control characters in the data.
- Compressing blank fields within the data, thereby increasing the transmission efficiency. The IBM 3780 does this automatically by replacing blank fields with a two-byte control sequence.
- Transmitting or receiving blocked data thereby increasing the transmission efficiency. The maximum number of records allowed per block is 216 in the non-transparent mode and 1 (optionally 6) in the transparent mode.
- Interleaving the data passed between the communications line and the input/output devices through the use of two 512-byte buffers, thereby increasing the overall transmission efficiency.
- Generating WACK (Wait-before-transmit ACKnowledgement) and TTD (Temporary Text Delay) control character sequences when it is temporarily unable to transmit or receive data.

HP 2780/3780 Emulator Capabilities

Except for the use of six-bit transcode, the HP 2780/3780 Emulator provides all of the capabilities mentioned above plus the following:

- When emulating an IBM 2780, it performs short-record truncation without the user having to supply EM control characters in the data.

- When emulating an IBM 2780, it can perform blank field compression.
- When emulating either an IBM 2780 or 3780, it provides input/output access to a greater variety of peripheral devices including disc and magnetic tape.
- When emulating an IBM 2780, it can block more than 7 records.

System Environment

The HP 2780/3780 Emulator is designed to operate under the control of MPE and requires the minimum HP 3000 Series II configuration plus one of the following MODEMs or its equivalent:

- A Bell System Type 201A3 Data Set for 2000 bit-per-second (bps) transmission via the public telephone (switched) network.
- A Bell System Type 201B3 Data Set for 2400 bps transmission via a private leased line.
- A Bell System Type 201C Data Set for 2400 bps transmission via either the public telephone (switched) network or a private leased line.
- A Bell System Type 208A Data Set for 4800 bps transmission via a private leased line.
- A Bell System Type 208B Data Set for 4800 bps transmission via the public telephone (switched) network.

The MODEM options recommended for use with the HP 2780/3780 Emulator are presented in Appendix B of the emulator reference manual.

Documentation

For a detailed description of the IBM 2780 refer to *Component Description: IBM 2780 Data Transmission Terminal* (IBM SRL GA27-3005).

For a detailed description of the IBM 3780 refer to *Component Description for the IBM 3780 Data Communication Terminal* (IBM SRL GA27-3063).

The HP 2780/3780 Emulator is described in the *HP 2780/3780 Emulator Reference Manual* (HP part number 30000-90047).

Example

Figure 7-2 shows an example of a batch job that communicates with a remote job processing system over the public telephone network. The job deck is a combination of card decks supplied by several users and the computer operator. The job transmits three input data sets to the remote processor and receives three list output data sets and one punched output data set in return.

In this example the remote job processing system is assumed to be similar to IBM HASP/360. For example, it is assumed that:

- the computer operator does *not* have to transmit a terminal ID when establishing the communications link
- the computer operator must transmit a "log-on" card image immediately followed by an EOT control character after the communications link has been established and before any other input/output activity is performed
- the computer operator must transmit a "log-off" card image immediately followed by an EOT control character just before terminating the communications link

By making minor alterations to the command sequence, the user can adapt this example to communicate with most any remote job processing system.

The example is based on the following assumptions:

Mode Of Operation: Batch job.

Device Emulated: IBM 2780.

Remote Processor: A remote job processing system that requires a /*$LOGON card immediately followed by an EOT control character at the start of the run and a /*$LOGOFF card at the end of the run.

Communications Link: Public telephone network. The computer operator of your HP 3000 Computer System must manually make the telephone connection with the remote processor.

Transmission Code: ASCII.

Input Code: ASCII.

Command File: Card reader ($STDINX, the standard job input device).

Input File: Card reader ($STDINX, the standard job input device).

List File: Line printer ($STDLIST, the standard job listing device).

Punch File: Any card punch in the system (in this example CP is assumed to be the device class name for all card punches in the system).

The job deck is as follows:

```
    :JOB username.acctname
①   :FILE PNCH;DEV=CP
②   :RJE ,,,*PNCH
③   #RJLINE 2780;CONNECT=DIAL, "247-7000";LINECODE=ASCII
④   #RJIN ;TRUNCATE=NO
    /*$LOGON
⑤   #RJEOD
```

Supplied by the computer operator.

```
⑥   #RJIN
    Input Data
    Set #1
⑦   #RJEOD
```

Supplied by user "A"

```
⑧   #RJIN
    Input Data
    Set #2
⑨   #RJEOD
```

Supplied by user "B"

```
⑩   #RJIN
    Input Data
    Set #3
⑪   #RJEOD
```

Supplied by user "C"

```
⑫   #RJOUT ,99;WAIT=5
⑬   #RJIN ;TRUNCATE=NO
    /*$LOGOFF
⑭   #RJEND
    :EOD
    :EOJ
```

Supplied by the computer operator.

Figure 7-2. Batch Remote Job Entry Example

The annotations are as follows:

(1) This command equates the device class name CP to the formal file designator PNCH.

(2) This command invokes the Emulator and defines the four Emulator files as follows:

Command File:	$STDINX (the standard job input device).
Input File:	$STDINX.
List File:	$STDLIST (the standard job listing device).
Punch File:	Any card punch in the system.

(3) This command states that an IBM 2780 is to be emulated, the transmission code is to be ASCII, the communications link is to be established by your computer operator manually dialing the remote processor (247-7000) via a MODEM, the maximum number of records per transmission block is to be 7 (non-transparent mode) or 2 (transparent mode), no terminal ID is to be transmitted, and no RIN is to be used.

(4) This pair of cards transmits an 80-column card image to the remote processor. The card image contains /*$LOGON in positions 1-8 and blanks (spaces) in positions 9-80.

(5) This card transmits an EOT control character to the remote processor, thus completing the transmission initiated by the preceding #RJIN command.

(6) This command transmits the first input data set to the remote processor. The input data set immediately follows the #RJIN command in the job deck. The #RJIN command states that the entire input data set is to be transmitted, no data compression is to be performed, the non-transparent mode is to be used, short-record truncation is to be performed, and the input data set is in ASCII format.

(7) This command transmits an EOT control character to the remote processor, thus completing the transmission initiated by the preceding #RJIN command.

(8) This command transmits the second input data set to the remote processor. The input data set immediately follows the #RJIN command in the job deck. The #RJIN command states that the entire input data set is to be transmitted, no data compression is to be performed, the non-transparent mode is to be used, short-record truncation is to be performed, and the input data set is in ASCII format.

(9) This command transmits an EOT control character to the remote processor, thus completing the transmission initiated by the preceding #RJIN command.

(10) This command transmits the third input data set to the remote processor. The input data set immediately follows the #RJIN command in the job deck. The #RJIN command states that the entire input data set is to be transmitted, no data compression is to be performed, the non-transparent mode is to be used, short-record truncation is to be performed, and the input data set is in ASCII format.

(11) This command transmits an EOT control character to the remote processor, thus completing the transmission initiated by the preceding #RJIN command.

(12) This command causes the Emulator to turn the line around and wait for output from the remote processor. The #RJOUT command states that 99 output data sets are expected. This number was chosen as a convenience so that the same #RJOUT card could be used in many different runs without having to be altered. The #RJOUT command will be terminated by the Emulator whenever 5 minutes elapses without anything being received from the remote processor.

(13) This pair of cards transmits an 80-column card image to the remote processor. The card image contains /*$LOGOFF in positions 1-9 and blanks (spaces) in positions 10-80.

(14) This command transmits an EOT control character to the remote processor, thus completing the transmission initiated by the preceding #RJIN command, and then terminates the HP 2780/3780 Emulator subsystem.

REAL-TIME PROGRAMMABLE CONTROLLER

The HP Real-Time Programmable Controller operates under the HP 2100 Real-Time Executive Core-Based Operating System (RTE-C) to provide multi-programmed, real-time input/output processing for the HP 3000 Series II. The controller consists of a small central processing unit (HP 2100), plug-in interface cards, a signal cable to connect the controller to the HP 3000 Series II, and all necessary software.

Because it is independently programmable under RTE-C, the controller gives the user efficient access to, and control over, a wide range of real-time input/output devices which are not normally connected to the HP 3000 Series II.

With the supplied software, all phases of implementing the real-time capability — from program development, to system configuration, to program control — can be accomplished on the HP 3000 Series II. From jobs or sessions on the HP 3000 Series II the user can:

- generate an RTE-C system
- transmit the RTE-C system to the controller
- assemble programs to run under RTE-C
- transmit 3000-relocated programs to the controller
- specify whether the transmitted programs should be dormant or scheduled in RTE-C
- transfer data between the 3000 and input/output devices running under RTE-C

The transmitting and receiving of data may be initiated from the RTE-C end as well as from the HP 3000 Series II.

Features of RTE-C

As a multiprogramming, real-time operating system, RTE-C provides features which allow the user to:

- assign a priority value for each program in the system
- run multiple programs simultaneously
- replace or load programs while others are running
- terminate or remove programs
- activate any program by operator request, program request, device interrupt, or time interval
- perform input/output with a simple READ/WRITE call referencing a logical unit and a core buffer of definable size

These features apply both to the stand-alone RTE-C system and to RTE-C as used in the HP Real-Time Programmable Controller environment.

Hardware Components

The HP Real-Time Programmable Controller consists of an interface kit (two interface cards, one for the 2100 and one for the 3000, plus an interconnecting cable) and the minimum hardware configuration required to support RTE-C, namely:

- an HP 2100 Computer, with 8K memory and Direct Memory Access (DMA)
- a Time Base Generator (TBG)
- a paper tape reader
- a teleprinter (TTY)

These hardware components are connected as shown in figure 7-3. As indicated, the interconnecting cable may be up to 250 feet (76.2m) long. Cables longer than 50 feet, however, will affect the transmission rate. A 50-foot cable permits transfers of 16-bit words at about 200,000 words per second whereas a 250-foot cable reduces the rate to about 175,000 words per second. All transfers to and from the HP 3000 Series II occur via a Direct Memory Access (DMA) channel in the HP 2100.

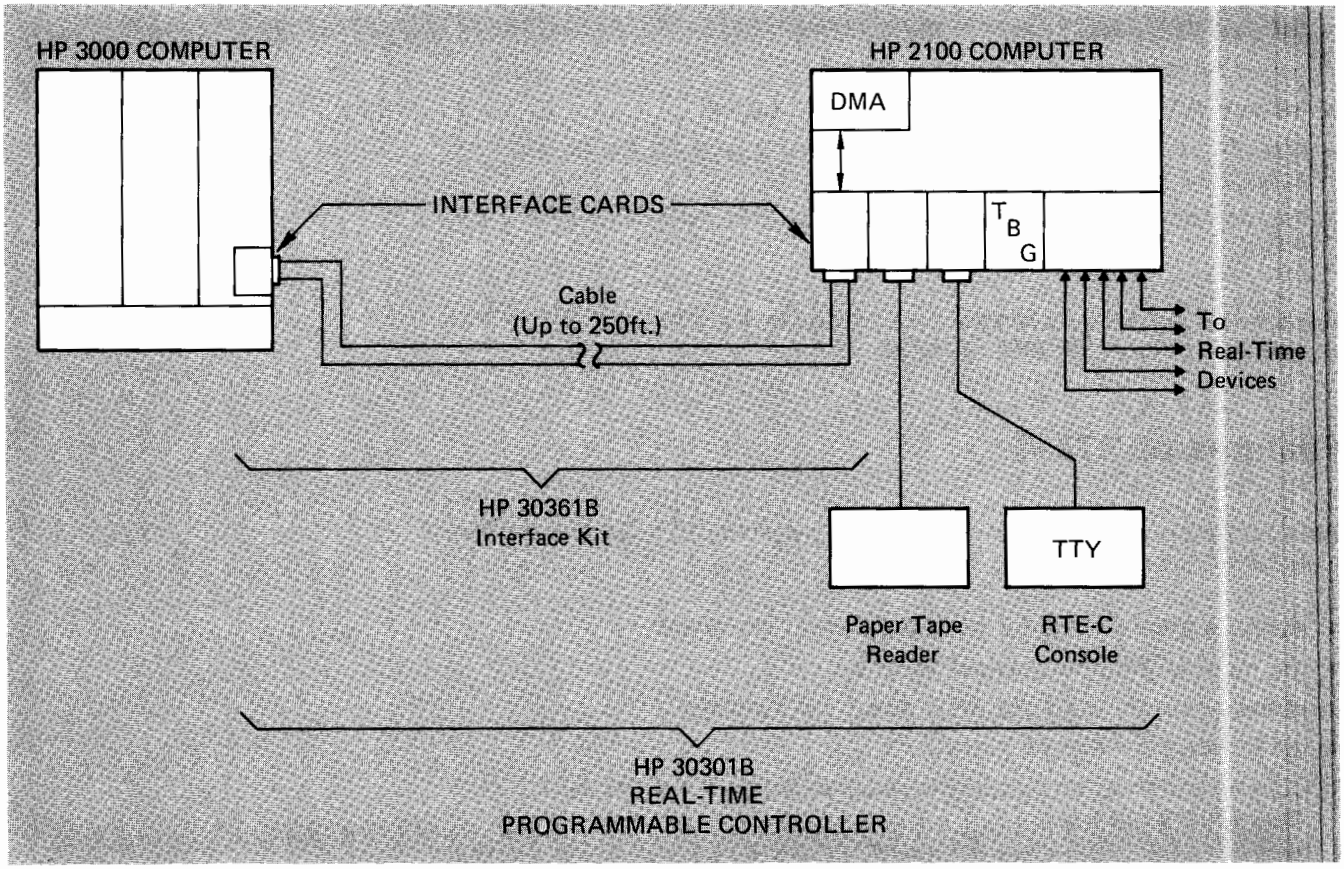


Figure 7-3. Real-Time Programmable Controller Hardware Components

Software Components

For the HP 3000 Series II, the supplied software includes:

- a procedure (DNLDSYS) used to transfer an absolute RTE-C system, generated in the HP 3000, to the HP 2100
- a procedure (DNLDUSER) used to transfer absolute user program files, created in the HP 3000, to the HP 2100
- a Cross Generator program (XRTCGEN) used to generate Cross Loader commands for generation of an absolute RTE-C system
- a diagnostic program (LINETEST) used with the HP 2100 program LTEST to test the interface link
- a Cross Assembler program which assembles HP 2100 assembly language programs on the HP 3000
- a Cross Loader program which is used to generate RTE-C and to convert assembled relocatable programs to absolute form for downloading

For the HP 2100, the supplied software includes:

- a Basic Binary Link Loader (BBLL) which receives the configured absolute RTE-C system sent across the link by the HP 3000 procedure DNLDSYS, loads it into memory, and starts it running
- various modules of RTE-C including a modified version of the absolute program loader (APLDR) which receives absolute user programs and program data sent across the link by the HP 3000 procedure DNLDUSER, builds ID segments, and optionally schedules programs for execution
- a diagnostic program (LTEST) used with the HP 3000 program LINETEST to test the interface link

Applications Software

Since the controller itself is basically a central processing unit, the user must write his application in two parts: a program to execute under MPE in the HP 3000 Series II and a program to execute under RTE-C in the controller. The application must be designed so that the HP 3000 program and the HP 2100 program work together as a program pair.

Typically, the controller program (independent of the HP 3000) controls input/output devices or instruments, reads or writes information to them, buffers any data collected, and transfers this data across the link to the HP 3000 portion of the application. Once the HP 3000 program receives the data, it may analyze it, list it, or write it to disc. The functions of each half of a program pair, however, depend solely on the application being implemented.

Programs written for the HP 3000 operate under the control of MPE. Data transfers across the link to the controller are achieved by calling the standard File System intrinsics FOPEN, FREAD, FWRITE, FCONTROL, and FCLOSE. The user can develop the HP 3000 programs in any language capable of calling the MPE File System intrinsics.

Programs written for the controller operate under the control of RTE-C. Input/output device operations (as well as controller-initiated data transfers to the HP 3000) are achieved by standard RTE-C calls. HP 2100 assembler language, as documented in the Cross Assembler reference manual (with modified NAM and ORB instructions as documented in the RTE-C reference manual), is used to write these 2100 application programs. The controller is truly programmable in that the user program which directs its actions may be changed by merely downloading a different user program.

Documentation

The HP Real-Time Programmable Controller is described in the *Real-Time Programmable Controller Reference and Application Manual* (HP part number 30000-90067).

The RTE-C operating system is described in the *Real-Time Executive Core-Based Software System Programming and Operating Manual* (HP part number 29101-93001). Because the software supplied with the HP Real-Time Programmable Controller performs system generation and program loading in a manner different from that described in the RTE-C reference manual, the information in that manual pertaining to RTE-C System Generation and the RTE-C Relocating Loader does not apply in the environment of the controller. The information pertaining to Real-Time Program Preparation applies primarily to stand-alone RTE-C systems; however, the portion dealing with the modification of the NAM and ORB statements does apply for the Real-Time Programmable Controller.

The Cross Assembler program is described in the *HP 3000 Cross Assembler for HP 2100 Computers Reference and Application Manual* (HP part number 03000-90047).

The Cross Loader program is described in the *HP 3000 Cross Loader for HP 2100 Computers Reference and Application Manual* (HP part number 03000-90107).

PROGRAMMABLE CONTROLLER

The HP Programmable Controller operates under the HP 2100 Basic Control System (BCS) to provide input/output processing for the HP 3000 Series II. The controller consists of a small central processing unit (HP 2100), plug-in interface cards, a signal cable to connect the controller to the HP 3000 Series II, and all necessary software.

Because it is independently programmable under BCS, the controller gives the user efficient access to, and control over, a wide range of input/output devices which are not normally connected to the HP 3000 Series II.

With the supplied software, all phases of implementing a programmable controller application — from program development to program control — can be accomplished on the HP 3000 Series II. From jobs or sessions on the HP 3000 Series II the user can:

- generate BCS
- assemble programs to run under BCS
- relocate 3000-assembled programs
- download 3000-relocated programs to the controller
- transfer data between the 3000 and input/output devices running under BCS

The transmitting and receiving of data may be initiated from the BCS end as well as from the HP 3000 Series II.

Hardware Components

The HP Programmable Controller consists of an HP 2100 Computer with 8K memory and Direct Memory Access (DMA), a paper tape reader, and an interface kit (two interface cards, one for the 2100 and one for the 3000, plus an interconnecting cable). These hardware components are connected as shown in figure 7-4. As indicated, the interconnecting cable may be up to 250 feet (76.2m) long. Cables longer than 50 feet, however, will affect the transmission rate. A 50-foot cable permits transfers of 16-bit words at about 200,000 words per second whereas a 250-foot cable reduces the rate to about 175,000 words per second. All transfers to and from the HP 3000 Series II occur via a Direct Memory Access (DMA) channel in the HP 2100.

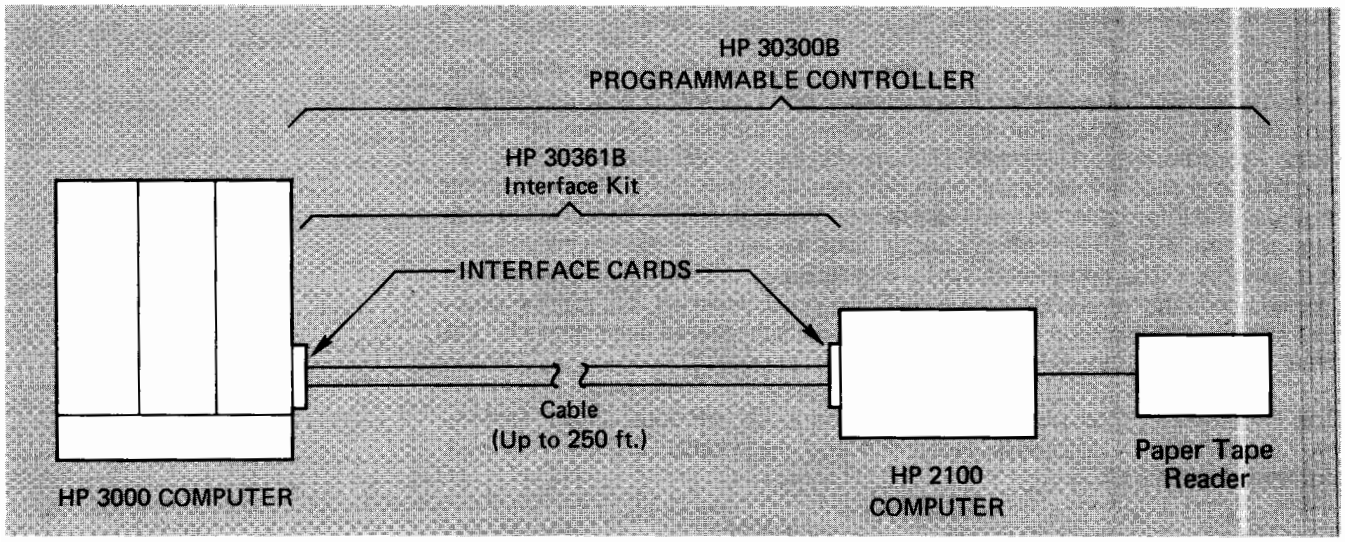


Figure 7-4. Programmable Controller Hardware Components

Software Components

For the HP 3000 Series II, the supplied software includes:

- a procedure (DOWNLOAD) used to transfer absolute user program files, created in the HP 3000, to the HP 2100
- a link driver (IOREM0) called by the MPE File System to read, write, set end-of-file/unusual condition, and obtain status of the link
- a diagnostic program (LINETEST) used with the HP 2100 program LTEST to test the interface link
- a Cross Assembler program which assembles HP 2100 assembly language programs on the HP 3000
- a Cross Loader program which is used to generate BCS and to convert assembled relocatable programs to absolute form for downloading.

For the HP 2100, the supplied software includes:

- a Basic Binary Link Loader (BBLL) which receives the absolute program files sent across the link by the HP 3000 procedure DOWNLOAD, loads them into memory, and sets them into execution
- a link driver (D.63) called by the BCS input/output control subroutine to read, write, clear, set end-of-file/unusual condition, and obtain status information from the link
- a diagnostic program (LTEST) used with the HP 3000 procedure LINETEST to test the interface link

Applications Software

Since the controller itself is basically a central processing unit, the user must write his application in two parts: a program to execute under MPE in the HP 3000 Series II and a program to execute under BCS in the controller. The application must be designed so that the HP 3000 program and the HP 2100 program work together as a program pair.

Typically, the controller program (independent of the HP 3000) controls input/output devices or instruments, reads or writes information to them, buffers any data collected, and transfers this data across the link to the HP 3000 portion of the application. Once the HP 3000 program receives the data, it may analyze it, list it, or write it to disc. The functions of each half of a program pair, however, depend solely on the application being implemented.

Programs written for the HP 3000 operate under the control of MPE. Data transfers across the link to the controller are achieved by calling the standard File System intrinsics FOPEN, FREAD, FWRITE, FCONTROL, and FCLOSE. The user can develop the HP 3000 programs in any language capable of calling the MPE File System intrinsics.

Programs written for the controller operate under the control of BCS. Input/output device operations (as well as controller-initiated data transfers to the HP 3000) are achieved by standard calls to the IOC subroutine of BCS. Note that since the link itself is a high-speed device, its driver is designed to work with the unbuffered version of IOC. HP 2100 assembler language, as documented in the Cross Assembler reference manual, is used to write these 2100 application programs. The controller is truly programmable in that the user program which directs its actions may be changed by merely downloading a different user program.

Documentation

The HP Programmable Controller is described in the *Programmable Controller Reference and Application Manual* (HP part number 30000-90066).

The BCS operating system is described in *Basic Control System* (HP part number 02116-9017).

The Cross Assembler program is described in the *HP 3000 Cross Assembler for HP 2100 Computers Reference and Application Manual* (HP part number 03000-90047).

The Cross Loader program is described in the *HP 3000 Cross Loader for HP 2100 Computers Reference Manual* (HP part number 03000-90107).

APPENDIX A: MPE-II Commands

All MPE-II commands are summarized below, grouped alphabetically and by 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.

Computer Museum

STANDARD CAPABILITY COMMANDS

Command	Function	When Issued
:ABORT	Aborts the current program.	B
:ALTSEC	Changes security provisions for a file.	J,S,B,P
:BASIC	Calls BASIC/3000 interpreter.	J,S
:BASICOMP	Compiles a BASIC/3000 program.	J,S
:BASICGO	Compiles, prepares, and executes 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
:DSLINE	Opens or closes communication line with DS/3000.	J,S
:EDITOR	Calls the EDITOR.	J,S
: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

Command	Function	When Issued
: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
:GETRIN	Acquires a global RIN.	J,S,B,P
:HELLO	Initiates a session. Acceptable for device recognition. Requires 1A capability class.	S
:JOB	Initiates a batch job. Requires BA capability class.	J
:LISTF	Lists descriptions of files.	J,S,B,P
: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
:RELEASE	Releases the security provisions of a file.	J,S,B,P
:REMOTE	Establishes communication between a local computer and a remote computer.	J,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
: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

Command	Function	When Issued
: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/3000 Segmenter.	J,S
:SETDUMP	Enables the MPE Stackdump Facility on abort.	J,S,B,P
:SETMSG	Disables or enables receipt of user or operator messages at standard list device.	J,S,B,P
:SHOWDEV	Reports status of input/output devices.	J,S,B,P
:SHOWIN	Reports status of input devicefiles.	J,S,B,P
:SHOWJOB	Displays job/session status.	J,S,B,P
:SHOWOUT	Reports status of output devicefiles.	J,S,B,P
:SHOWTIME	Displays current date and time-of-day.	J,S,B,P
: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

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
–COPY	Copies an RBM or segment from one USL to another.	J,S
–EXIT	Exits from Segmenter, returning control to MPE-II 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	Lists the segments in an SL.	J,S
–LISTUSL	Lists the RBM's in a USL.	J,S
–NEWSEG	Changes the segment name of an RBM.	J,S
–PREPARE	Prepares RBM's from a USL into a program file.	J,S
–PURGERBM	Deletes one or more RBM's 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

Command	Function	When Issued
-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
:LISTACCT	Lists attributes of an account.	J,S,B,P
:NEWACCT	Creates a new account.	J,S,B,P
:PURGEACCT	Deletes an account.	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	List 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	Deletes a group from log-on account.	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,B,P
:QUANTUM	Changes a circular subqueue time-quantum.	J,S,B,P
:RESTORE	Reads back into the system, onto disc, any file, or set of files, that has been stored off-line (on tape) by :STORE or SYSDUMP.	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 onto magnetic tape any disc file or fileset.	J,S,B,P
:SWITCHLOG	Closes current Log File, opens a new Log File, and treats this as current Log File.	J,S,B,P
:SYSDUMP	Copies MPE-II to tape.	J,S

APPENDIX B: MPE-II Intrinsics

All MPE-II intrinsics (system procedures) available to the user are summarized below, listed alphabetically. If any special capability is required to call a particular intrinsic, that capability is noted.

Intrinsic	Function
ACCEPT	Accepts (and completes) the request received by the preceding GET intrinsic call.
ACTIVATE	Activates a process. (Requires PH capability.)
ADJUSTUSLF	Moves information block on a USL file.
ALTDSEG	Changes 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.
COMMAND	Executes an MPE-II 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.)

Intrinsic	Function
DMOVEOUT	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.)
FCHECK	Requests details about file input/output errors.
FCLOSE	Closes a file.
FCONTROL	Performs various control operations on a file or terminal device.
FGETINFO	Requests access and status information about a file.
FLOCK	Dynamically locks a file.
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 RIN's from allocation to a job.
FRELATE	Declares a file pair interactive or duplicative.
FRENAME	Renames a disc file.
FSETMODE	Activates or de-activates 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.
FWRITE	Writes a logical record to a sequential file.

Intrinsic	Function
FWRITEDIR	Writes a logical record to a direct-access disc file.
FWRITELABEL	Writes a user file label.
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.
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.
PCHECK	Returns an integer code specifying the completion status of the most recently executed slave program-to-program intrinsic.

Intrinsic	Function
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.
PRINTOP	Prints a character string on operator's console.
PROCTIME	Returns a process' accumulated central-processor time.
PTAPE	Accepts input from tapes not containing X-OFF control characters.
PWRITE	Sends a block of data to the remote slave program.
QUIT	Aborts a process.
QUITPROG	Aborts a program.
READ	Reads an ASCII string from an input device.
RECEIVEMAIL	Receives mail from another process. (Requires PH capability.)
RESETCONTROL	Re-sets a terminal to accept a CONTROL-Y signal.
SEARCH	Searches an array for a specified entry or name.
SENDMAIL	Sends mail to another process. (Requires PH capability.)
SETJCW	Sets bits in job control word.
SUSPEND	Suspends a process. (Requires PH capability.)

Intrinsic	Function
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.
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 system trap.
ZSIZE	Changes size of Z-DB area.

APPENDIX C: HP 3000 Series II Documentation

This appendix summarizes all of the available user documentation for the HP 3000 Series II Computer System. Figure C-1 does this in the form of a block diagram. The remainder of the appendix presents the full title, HP part number, and a brief description of each document. For ordering information see your HP Sales Representative.

Computer Museum

OPERATING SYSTEM MANUALS

MPE Commands Reference Manual (30000-90009)

Describes how to communicate with MPE-II using operating system commands entered through an interactive terminal or as part of a batch job stream.

MPE Intrinsics Reference Manual (30000-90010)

Describes how to communicate with MPE-II programmatically using intrinsic (system procedure) calls. It also describes the capabilities that are available to users who have been assigned special attributes by the System Manager/System Supervisor.

MPE Segmenter Reference Manual (30000-90011)

Describes how to manage User Subprogram Library (USL) files, build and manage Segmented Library (SL) files, and alter the segmentation of a program using the MPE-II Segmenter.

MPE Debug/Stack Dump Reference Manual (30000-90012)

Describes how to debug programs using the debug and stack dump features of MPE-II.

MPE System Utilities Manual (30000-90044)

Describes how to use MPE-II system utilities which perform such functions as copying disc files to magnetic tape for later retrieval, printing analytical listings of main memory dumps, modifying and listing the content of disc files, and displaying or modifying the content of Segmented Library (SL) files.

Index to MPE Reference Documents (30000-90045)

The master index for the entire set of MPE-II operating system user manuals.

Error Messages and Recovery Manual (30000-90015)

Summarizes the probable causes and recommended recovery actions for operating system and subsystem error messages.

System Manager/System Supervisor Manual (30000-90014)

Describes the commands and capabilities available to the system administrators (System Manager, System Supervisor, and Account Manager) of an HP 3000 Series II.

Console Operator's Guide (30000-90013)

Describes the commands and functional capabilities available to the Console Operator of an HP 3000 Series II. It also describes the operating instructions for the machine-resident microdiagnostic programs.

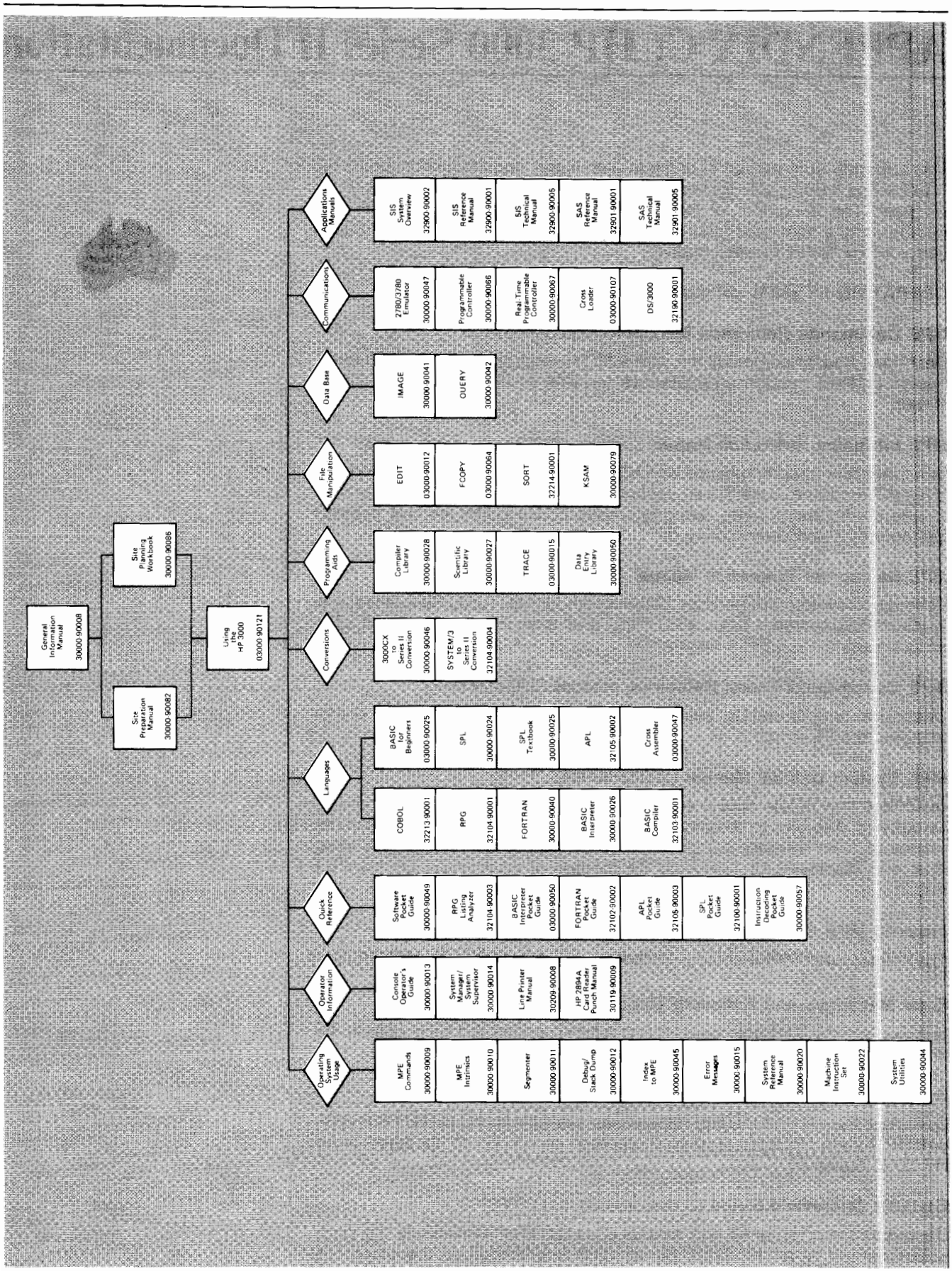


Figure C-1. HP 3000 Series II Documentation

HP 3000CX to HP 3000 Series II Program Conversion Guide (30000-90046)

Describes how to convert programs and data files created on an HP 3000CX so that they can be used on an HP 3000 Series II. Covers MPE, FORTRAN, BASIC, SPL, and IMAGE.

IBM SYSTEM/3 to HP 3000 Conversion Guide (32104-90004)

Describes how to convert programs and data files created on an IBM SYSTEM/3 so that they can be used on an HP 3000 Series II.

Line Printer Operating and Programming Manual (30209-90008)

Describes the operating instructions and programming information for the HP 2607A, HP 2613A, HP 2617A, and HP 2618A line printer subsystems available with the HP 3000 Series II Computer System.

HP 2894A Card Reader Punch Operating Manual (30119-90009)

Describes the operating and programming information for the HP 2894A Card Reader Punch.

HP 3000 Software Pocket Guide (30000-90049)

Summarizes all MPE-II commands, intrinsics (system procedures), and File System error messages, all Text Editor, File Copier, Sort/Merge, and 2780/3780 Emulator commands, and all language compiler subsystem commands. It also summarizes the ASCII character set and collating sequence, the HP 3000 Series II machine instruction set, the condition codes and status register, and the interrupt/trap segments and parameters.

MACHINE REFERENCE MANUALS

Site Preparation Manual (30000-90082)
Site Planning Workbook (30000-90086)

These two manuals help you prepare your site for the installation of an HP 3000 Series II Computer System.

System Reference Manual (30000-90020)

Provides a detailed technical description of the architecture and principles of operation of the HP 3000 Series II Computer System.

Machine Instruction Set Reference Manual (30000-90022)

Describes all of the machine instructions of the HP 3000 Series II Computer System.

Instruction Decoding Pocket Guide (30000-90057)

Summarizes all of the machine instructions of the HP 3000 Series II Computer System.

PROGRAMMING LANGUAGE MANUALS

COBOL/3000 Reference Manual (32213-90001)

Describes the constructs and use of the COBOL/3000 language.

RPG/3000 Compiler Reference and Application Manual (32104-90001)

Describes the constructs and use of the RPG/3000 language.

RPG Listing Analyzer (32104-90003)

A column and field template which makes it easier to read RPG listings.

FORTRAN Reference Manual (30000-90040)

Describes the constructs and use of the FORTRAN/3000 language.

FORTRAN Pocket Guide (32102-90002)

Summarizes the constructs of the FORTRAN/3000 language.

BASIC Interpreter Reference Manual (30000-90026)

Describes the constructs and use of the BASIC/3000 language.

BASIC/3000 Compiler Reference Manual (32103-90001)

Describes how to compile a BASIC/3000 program.

BASIC for Beginners (03000-90025)

Describes the following five functional elements of the BASIC/3000 language:

- Calculating values
- Transferring control
- Looping
- Functions and subroutines
- Strings and files

BASIC/3000 Interpreter Pocket Guide (03000-90050)

Summarizes the constructs of the BASIC/3000 language.

Systems Programming Language Reference Manual (30000-90024)

Describes the constructs and use of the HP 3000 Systems Programming Language (SPL/3000).

Systems Programming Language Textbook (30000-90025)

Describes the constructs and use of SPL/3000 in a tutorial manner. Designed for self-study, each chapter concludes with an examination.

SPL Pocket Guide (32100-90001)

Summarizes the constructs of the SPL/3000 language.

APL Reference Manual (32105-90002)

Describes the constructs and use of the APL/3000 language.

APL Pocket Guide (32105-90003)

Summarizes the constructs of the APL/3000 language.

Cross Assembler for 2100 Computers Reference and Application Manual (03000-90047)

Describes how to assemble HP 2100 Assembler Language programs on an HP 3000.

ADDITIONAL MANUALS

EDIT/3000 Reference Manual (03000-90012)

Describes the constructs and use of the HP 3000 Text Editor.

FCOPY Reference Manual (03000-90064)

Describes the constructs and use of the HP 3000 File Copier.

Sort/3000 Reference Manual (32214-90001)

Describes the constructs and use of the HP 3000 Sort/Merge program.

KSAM Reference Manual (30000-90079)

Describes the constructs and use of HP's Keyed Sequential Access Method for accessing files indexed with keys.

HP 3000 Compiler Library Reference Manual (30000-90028)

Describes what each input/output formatting, mathematical, and utility procedure in the HP 3000 Compiler Library does and how to call them.

HP 3000 Scientific Library Reference Manual (30000-90027)

Describes what each mathematical, statistical, and utility procedure in the HP 3000 Scientific Library does and how to call them.

Computer Museum

HP 3000 Symbol Trace Reference Manual (03000-90015)

Describes how to debug programs using the HP 3000 Symbol Trace program.

IMAGE Reference Manual (30000-90041)

Describes how to create, maintain, and use an IMAGE/3000 data base.

QUERY Reference Manual (30000-90042)

Describes how to access an IMAGE/3000 data base from an interactive terminal using the QUERY/3000 program.

2780/3780 Emulator Reference Manual (30000-90047)

Describes how to transfer data between an HP 3000 and a variety of remote processors in a full multiprogramming environment over the public telephone (switched) network or a private leased line using the HP 30130 2780/3780 Emulator Subsystem. The Emulator makes the HP 3000 appear to the remote processor as either an IBM 2780 or 3780 Data Transmission Terminal.

Programmable Controller Reference and Application Manual (30000-90066)

Describes how to program the hardwired communications link between an HP 3000 and an HP 2100 (with a BCS operating system environment) provided by the HP 30300 Programmable Controller.

Real-Time Programmable Controller Reference and Application Manual (30000-90067)

Describes how to program the hardwired communications link between an HP 3000 and an HP 2100 (with an RTE-C operating system environment) provided by the HP 30301 Real-Time Programmable Controller.

HP 3000 Cross Loader for HP 2100 Computers Reference and Application Manual (03000-90107)

Describes how to do the following on an HP 3000 using the Cross Loader:

- Perform program relocation and linking for HP 2100 programs
- Generate HP 2100 BCS operating systems
- Transfer the resultant output of the above two operations from the HP 3000 to an HP 2100 over a communications link

DS/3000 Reference Manual (32190-90001)

Describes how a set of software enhancements to MPE provides direct communication between several HP 3000 computer systems. The connection can be made over hardwired (coaxial cable) lines or over public telephone lines.

Using the HP 3000: A Guide for the Terminal User (03000-90121)

An introduction to the use of the HP 3000 Series II. Covers terminal characteristics, initiating and terminating an interactive session, some MPE-II File System commands, the HP 3000 Text Editor, FORTRAN/3000, COBOL/3000, and the BASIC/3000 Interpreter.

Data Entry Library Reference Manual (30000-90050)

Describes how to do the following using the HP 3000 Data Entry Library (DEL/3000):

- Create forms interactively from an HP 2640/2644 Terminal and store them in the system
- Display, modify, and delete forms interactively from an HP 2640/2644 Terminal
- Call DEL/3000 procedures from within user programs to display the forms on a terminal and allow anyone running these user programs to enter data on the form images, edit this data, and store it in data files in the formats established by the user programs.

Student Information System — System Overview (32900-90002)

Briefly describes the features and suggested applications of the Student Information System (SIS/3000) data storage and retrieval system which is designed for use in educational institutions.

Student Information System Reference Manual (32900-90001)

Describes how to implement an educational information system using the three SIS/3000 subsystems (Attendance Accounting, Mark Reporting, and Family Information Reporting).

Student Information System — Technical Manual (32900-90005)

Describes the internal maintenance specifications for the HP-supplied SIS/3000 software. This manual is necessary only if your application requires that HP-supplied software be modified. Note that modified HP software is not covered by the warranty.

Student Assignment System Reference Manual (32901-90001)

Describes how to implement a class scheduling system for educational institutions using the Student Assignment System (SAS/3000).

Student Assignment System — Technical Manual (32901-90005)

Describes the internal maintenance specifications for the HP-supplied SAS/3000 software. This manual is necessary only if your application requires that HP-supplied software be modified. Note that modified HP software is not covered by the warranty.

APPENDIX D: HP 3000 Series II Machine Instructions

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-justifiy
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 number
EXIT	Exit from procedure	PSDB	Pseudo interrupt disable
HALT	Halt	PSEB	Pseudo interrupt enable
IXIT	Interrupt exit	RSW	Read switch register
LLBL	Load label	SCAL	Subroutine call
LLSH	Linked list search	SXIT	Exit from subroutine
LOCK	Lock resource	UNLK	Unlock resource
PAUS	Pause, interruptable	XEQ	Execute stack word
PCAL	Procedure call		

I/O INSTRUCTIONS

CIO	Control I/O, direct	SIN	Set interrupt
CMD	Send command to module, direct	SIO	Start I/O, block transfer
RIO	Read I/O, direct	SMSK	Set device mask
RMSK	Read device mask	TIO	Test I/O, direct
SED	Set enable/disable external interrupts	WIO	Write I/O, direct

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		Decimal Arithmetic	
EADD	Add	ADDD	Decimal add
ECMP	Compare	CMPD	Decimal compare
EDIV	Divide	CVAD	ASCII to decimal conversion
EMPY	Multiply	CVBD	Binary to decimal conversion
ENEG	Negate	CVDA	Decimal to ASCII conversion
ESUB	Subtract	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

INDEX

Access Modes, MPE File 23
Account Manager 20
Account Resource Usage Limits 21
Accounting Facility, MPE 21
Addressing, Relative
  DB-plus 8
  Q-minus 8
  Q-plus 8
  S-minus 8
APL 39

Back-Up, System 27
BASIC 34
Batch Processing 17
BCS Programmable Controller 86

Capability Sets, MPE 26
Central Data Bus 6
Central Processing Unit 5
COBOL 30
Code Segment Table 9
Code Segments 7, 9
Commands, MPE ..... 18, Appendix A
Compiler Library 47
CPU 5
CST 9

Data Bases, KSAM 52
Data Bases, IMAGE 58
Data Entry Library 49
Data Segments 9
Data Sets, KSAM 52
Data Sets, IMAGE 58
Data Service Priority 13
Data Stacks 7
DB Register 7
Device Reference Table 13
Devices, Peripheral 14
Direct I/C 12
Distributed Systems 71
DIO 12
DRT 13

EDIT 43
Editor, Text 43
Emulator, 2780/3780 71
Errors, Command 18
Errors, Intrinsic 19

Fault Control Memory 11
FCOPY 46
File Copier 46
File System, MPE 22
FORTRAN 33

Groups 21

IMAGE 57
Input/Output Processor 2, 5
Instructions, Machine ..... 10, Appendix D
Interactive Processing 16
Interrupts 13
Intrinsics, MPE ..... 19, Appendix B
IOP 2, 5
IOP Bus 6

Jobs, Batch 17

KSAM 52

Lockwood, MPE File 22
Logging Facility, MPE 26

Memory, Main 11
Memory, Virtual 9
Microcode 11
Modification of MPE 27
Multiplexer Channel 3, 6, 12
Multiprogramming 17

Passwords, MPE File 22
Peripheral Devices 14
Processes 24
Program-to-Program Communications 72
Programmable Controller 86
Programmable Controller, Real-Time 82
Programmed I/O 12

Q Register 7
QUERY 64

RA Register 8
RB Register 8
RC Register 8
RD Register 8
Recovery, System 27
Recursion, Subprogram 7
Re-entrant Code 7
Registers 9
Relative Addressing
  DB-plus 8
  Q-minus 8
  Q-plus 8
  S-minus 8
Remote Command Processing 71
Remote File Access 72
RPG 31
RTE-C Programmable Controller 82

S Register 7
Scheduling 24
Scientific Library 48
Security, MPE File 22
Segmentation 9
Segmenter, MPE 25
Segmenter, Commands ..... Appendix A
Segments, Code 7, 9
Segments, Data 9
Selector Channel 3, 6, 13
Sessions 18
SIO 12
Sort-Merge 47
SPL 36
Spooling 17
Stacks, Data 7
Start-Up of MPE 27
Stream Facility 18
Swapping 9
System Administrators 20
System Manager 20
System Supervisor 20
Systems Programming Language 36

Text Editor 43
"Top of Stack" 7
TOS 7
Trace Facility 48
"Trap Procedure" 19

User Types, MPE File 23

Virtual Memory 9

Z Register 7

HEWLETT hp PACKARD

SALES & SERVICE OFFICES

AFRICA, ASIA, AUSTRALIA

ANGOLA
Telectra
Empresa Técnica de Equipamentos Eléctricos, S.A.R.L.
R. Barbosa Rodrigues, 42-I°DT.°
Caixa Postal, 6487
Luanda
Tel: 35515/6
Cable: TELECTRA Luanda

AUSTRALIA
Hewlett-Packard Australia Pty. Ltd.
31-41 Joseph Street
Blackburn, Victoria 3130
P.O. Box 36
Doncaster East, Victoria 3109
Tel: 89-6351
Telex: 31-024
Cable: HEWPARD Melbourne

Hewlett-Packard Australia Pty. Ltd.
31 Bridge Street
Pymble
New South Wales, 2073
Tel: 449-6566
Telex: 21561
Cable: HEWPARD Sydney

Hewlett-Packard Australia Pty. Ltd.
153 Greenhill Road
Parkside, S.A., 5063
Tel: 272-5911
Telex: 82536 ADEL
Cable: HEWPARD ADELAIDe

Hewlett-Packard Australia Pty. Ltd.
141 Stirling Highway
Nedlands, W.A. 6009
Tel: 86-5455
Telex: 93859 PERTH
Cable: HEWPARD PERTH

Hewlett-Packard Australia Pty. Ltd.
121 Wollongong Street
Fyshwick, A.C.T. 2609
Tel: 95-2733
Telex: 62650 Canberra
Cable: HEWPARD CANBERRA

Hewlett Packard Australia Pty. Ltd.
5th Floor
Teachers Union Building
495-499 Boundary Street
Spring Hill, 4000 Queensland
Tel: 229-1544
Cable: HEWPARD Brisbane

GUAM
Medical/Pocket Calculators Only
Guam Medical Supply, Inc.
Jay Ease Building, Room 210
P.O. Box 8947
Tamuning 96911
Tel: 646-4513
Cable: EARMED Guam

HONG KONG
Schmidt & Co.(Hong Kong) Ltd.
P.O. Box 297
Connaught Centre
39th Floor
Connaught Road, Central
Hong Kong
Tel: H-255291-5
Telex: 74766 SCHMC HX
Cable: SCHMIDTCO Hong Kong

INDIA
Blue Star Ltd.
Kasturi Buildings
Jamshedji Tata Rd.
Bombay 400 020
Tel: 29 50 21
Telex: 2156
Cable: BLUEFROST

Blue Star Ltd.
Sahas
414/2 Vir Savarkar Marg
Prabhadevi
Bombay 400 025
Tel: 45 78 87
Telex: 4093
Cable: FROSTBLUE

Blue Star Ltd.
Band Box House
Prabhadevi
Bombay 400 025
Tel: 45 73 01
Telex: 3751
Cable: BLUESTAR

Blue Star Ltd.
14/40 Civil Lines
Kanpur 208 001
Tel: 6 88 82
Telex: 292
Cable: BLUESTAR

Blue Star Ltd.
7 Hare Street
P.O. Box 506
Calcutta 700 001
Tel: 23-0131
Telex: 7655
Cable: BLUESTAR

Blue Star Ltd.
7th & 8th Floor
Bhandari House
91 Nehru Place
New Delhi 110024
Tel: 634770 & 635166
Telex: 2463
Cable: BLUESTAR

Blue Star Ltd.
Blue Star House
11/11A Magarath Road
Bangalore 560 025
Tel: 55668
Telex: 430
Cable: BLUESTAR

Blue Star Ltd.
Meeakshi Mandiran
xxx/1678 Mahatma Gandhi Rd.
Cochin 682 016
Tel: 32069, 32161, 32282
Telex: 046-514
Cable: BLUESTAR

Blue Star Ltd.
1-1-117/1
Sarojini Devi Road
Secunderabad 500 003
Tel: 70126, 70127
Cable: BLUEFROST
Telex: 459

Blue Star Ltd.
2/34 Kodambakkan High Road
Madras 600034
Tel: 82056
Telex: 041-379
Cable: BLUESTAR

Blue Star Ltd.
Nathraj Mansions
2nd Floor Bistupur
Jamshedpur 831 001
Tel: 7383
Cable: BLUESTAR
Telex: 240

INDONESIA
BERCA Indonesia P.T.
P.O. Box 496
1st Floor JL. Gikini Raya 61
Jakarta
Tel: 56038, 40369, 49886
Telex: 42895
Cable: BERCACON

BERCA Indonesia P.T.
63 JL. Raya Gubeng
Surabaya
Tel: 44309

ISRAEL
Electronics & Engineering Div. of Motorola Israel Ltd.
17, Kremenetski Street
P.O. Box 25016
Tel-Aviv
Tel: 38973
Telex: 33569
Cable: BASTEL Tel-Aviv

JAPAN
Yokogawa-Hewlett-Packard Ltd.
Ohashi Building
59-1 Yoyogi 1-Chome
Shibuya-ku, **Tokyo** 151
Tel: 03-370-2281/92
Telex: 232-2024YHP
Cable: YHPMARKET TOK 23-724

Yokogawa-Hewlett-Packard Ltd.
Seiko Ibaraki Building
2-8 Kasuga 2-chome, Ibaraki-shi
Osaka,567
Tel: (0726) 23-1641
Telex: 5332-385 YHP OSAKA

Yokogawa-Hewlett-Packard Ltd.
Nakamo Building
24 Kami Sasajima-cho
Nakamura-ku, **Nagoya**, 450
Tel: (052) 571-5171

Yokogawa-Hewlett-Packard Ltd.
Tamagawa Building
2-24-1 Tsuruya-cho
Kanagawa-ku
Yokohama, 221
Tel: 045-312-1252
Telex: 382-3204 YHP YOK

Yokogawa-Hewlett-Packard Ltd.
Mito Mitsu Building
105, Chome-1,San-no-maru
Mito, Ibaragi 310
Tel.: 0292-25-7470

Yokogawa-Hewlett-Packard Ltd.
Inoue Building
1348-3, Asahi-cho, 1-chome
Atsugi, Kanagawa 243
Tel: 0462-24-0452

Yokogawa-Hewlett-Packard Ltd.
Kimura Building
3rd Floor 20
2-chome, Tsukuba
Kumagaya, Saitama 360
Tel: 0485-24-6563

KENYA
Technical Engineering Services (E.A.)Ltd.
P.O. Box 18311
Nairobi
Tel: 557726/556762
Cable: PROTON

Medical Only
International Aeradio(E.A.)Ltd.,
P.O. Box 19012
Nairobi Airport
Nairobi
Tel: 336055/56
Telex: 22201/22301
Cable: INTAERIO Nairobi

KOREA
Samsung Electronics Co., Ltd.
20th Fl. Dongbang Bldg.250,2-KA
C.P.O. Box 2775
Taepyung-Ro, Chung-Ku
Seoul
Tel: (23) 6811
Telex: 22575
Cable: ELEKSTAR Seoul

MALAYSIA
Teknik Mutu Sdn. Bhd.
2 Lorong 13/6A
Section 13
Petaling Jaya,**Selangor**
Tel: 54994/54916
Telex: MA 37605

Protel Engineering
P.O. Box 1917
Lot 259, Satok Road
Kuching, **Sarawak**
Tel: 2400
Cable: PROTEL ENG

MOZAMBIQUE
A.N. Goncalves, Lta.
162, 1° Apt. 14 Av. D. Luis
Caixa Postal 107
Lourenco Marques
Tel: 27091, 27114
Telex: 6-203 NEGON Mo
Cable: NEGON

NEW ZEALAND
Hewlett-Packard (N.Z.) Ltd.
P.O. Box 9443
Courtney Place
Wellington
Tel: 877-199
Telex: NZ 3839
Cable: HEWPACK Wellington

Hewlett-Packard (N.Z.) Ltd.
Pakuranga Professional Centre
267 Pakuranga Highway
Box 51092
Pakuranga
Tel: 569-651
Telex: NZ 3839
Cable: HEWPACK Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
Scientific Division
79 Carlton Gore Rd., Newmarket
P.O. Box 1234
Auckland
Tel: 75-289
Telex: 2958 MEDISUP
Cable: DENTAL Auckland

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
P.O. Box 1994
147-161 Tory St.
Wellington
Tel: 850-799
Telex: 3858
Cable: DENTAL, Wellington

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
P.O. Box 309
239 Stanmore Road
Christchurch
Tel: 892-019
Cable: DENTAL, Christchurch

Analytical/Medical Only
Medical Supplies N.Z. Ltd.
303 Great King Street
P.O. Box 233
Dunedin
Tel: 88-817
Cable: DENTAL, Dunedin

NIGERIA
The Electronics Instrumentations Ltd.
N6B/770 Oyo Road
Oluseun House
P.M.B. 5402
Ibadan
Tel: 61577
Telex: 31231 TEIL Nigeria
Cable: THETEIL Ibadan

The Electronics Instrumentations Ltd.
144 Agege Motor Road, Mushin
P.O. Box 6645
Lagos
Cable: THETEIL Lagos

PAKISTAN
Mushko & Company, Ltd.
Oosman Chambers
Abdullah Haroon Road
Karachi-3
Tel: 511027, 512927
Telex: KR894
Cable: CDOPERATOR Karachi

Mushko & Company, Ltd.
38B, Satellite Town
Rawalpindi
Tel: 41924
Cable: FEMUS Rawalpindi

PHILIPPINES
The Online Advanced Systems Corporation
6th Floor, Yujui Co. Bldg.
560 Quintin Paredes St.
Binondo, **Manila**
Tel: 48-71-49, 48-68-53
In Makati 85-35-81, 85-34-91
Telex: 3274 ONLINE

RHODESIA
Field Technical Sales
45 Kelvin Road North
P.O. Box 3458
Saliabury
Tel: 705231 (5 lines)
Telex: RH 4122

SINGAPORE
Hewlett-Packard Singapore (Pte.) Ltd.
3841F, Jalan Bukit Merah
Blk. 2, 6th Floor, Jalan
Redhill Industrial Estate
Alexandra P.O. Box 58,
Singapore 3
Tel: 633022
Telex: HPSG RS 21486
Cable: HEWPACK, Singapore

SOUTH AFRICA
Hewlett-Packard South Africa (Pty.), Ltd.
Private Bag Wendywood
Sandton, Transvaal 2144
Hewlett-Packard Center
Daphne Street, Wendywood,
Sandton, Transvaal 2144
Tel: 802-104018
Telex: 8-4782
Cable: HEWPACK JOHANNESBURG

Service Department
Hewlett-Packard South Africa (Pty.), Ltd.
P.O. Box 39325
Gramley, Sandton, 2018
451 Wynberg Extension 3,
Sandton, 2001
Tel: 636-8188/9
Telex: 8-2391

Hewlett-Packard South Africa (Pty.), Ltd.
P.O. Box 120
Howard Place, Cape Province, 7450
Pine Park Center, Forest Drive,
Pinelands, Cape Province, 7405
Tel: 53-7955 thu 9
Telex: 57-0006

Service Department
Hewlett-Packard South Africa (Pty.), Ltd.
P.O. Box 37099
Overport, Durban 4067
Braby House
641 Ridge Road
Durban, 4001
Tel: 88-7478
Telex: 6-7954

TAIWAN
Hewlett-Packard Far East Ltd.,
Taiwan Branch
39 Chung Shiao West Road
Sec. 1, 7th Floor
Taipei
Tel: 3819160-4
Telex: 21824 HEWPACK
Cable: HEWPACK TAIPEI

Hewlett-Packard Far East Ltd.
Taiwan Branch
68-2, Chung Cheng 3rd. Road
Kaohsiung
Tel: (07) 242318-Kaohsiung

Analytical Only
San Kwang Instruments Co., Ltd.,
No. 20, Yung Sui Road
Taipei
Tel: 3715171-4 (5 lines)
Telex: 22894 SANKWANG
Cable: SANKWANG TAIPEI

TANZANIA
Medical Only
International Aeradio (E.A.), Ltd.
P.O. Box 861
Dar es Salaam
Tel: 21251 Ext. 265
Telex: 41030

THAILAND
UNIMESA Co., Ltd.
Elcom Research Building
Bangjak Sukumvit Ave.
Bangkok
Tel. 932387, 930338
Cable: UNIMESA Bangkok

UGANDA
Medical Only
International Aeradio(E.A.), Ltd.,
P.O. Box 2577
Kampala
Tel: 54388
Cable: INTAERIO Kampala

ZAMBIA
R.J. Tilbury (Zambia) Ltd.
P.O. Box 2792
Lusaka
Tel: 73793
Cable: ARJAYTEE, Lusaka

OTHER AREAS NOT LISTED, CONTACT:
Hewlett-Packard Intercontinental
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1267
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-8493

CANADA

ALBERTA
Hewlett-Packard (Canada) Ltd.
11620A - 168 Street
EdmontonT5M 3T9
Tel: (403) 452-3670
TWX: 610-831-2431 EDTH

Hewlett-Packard (Canada) Ltd.
915-42 Ave S.E., Suite 102
Calgary T2G 1Z1
Tel: (403) 287-1672
Twx: 610-821-6141

BRITISH COLUMBIA
Hewlett-Packard (Canada) Ltd.
837 E. Cordova Street
Vancouver V6A 3R2
Tel: (604) 254-0531
TWX: 610-922-5059 VCR

MANITOBA
Hewlett-Packard (Canada) Ltd.
513 Century St.
St. James
Winnipeg R3H 0L8
Tel: (204) 786-7581
TWX: 610-671-3531

NOVA SCOTIA
Hewlett-Packard (Canada) Ltd.
800 Windmill Road
P.O. Box 9331
Dartmouth, B2Y 3Z6
Tel: (902) 469-7820
TWX: 610-271-4482 HFX

ONTARIO
Hewlett-Packard (Canada) Ltd.
1020 Morrison Dr.
Ottawa K2H 8K7
Tel: (613) 820-6483
TWX: 610-513-1636

Hewlett-Packard (Canada) Ltd.
6877 Goreway Drive
Mississauga L4V 1M8
Tel: (416) 678-9430
TWX: 610-492-4246

QUEBEC
Hewlett-Packard (Canada) Ltd.
275 Hymus Blvd.
Pointe Claire H9R 1G7
Tel: (514) 697-4232
TWX: 610-422-3022
TLX: 05-821521 HPCL

FOR CANADIAN AREAS NOT LISTED:
Contact Hewlett-Packard (Canada) Ltd. in Mississauga.

CENTRAL AND SOUTH AMERICA

ARGENTINA
Hewlett-Packard Argentina S.A.
Av. Leandro N. Alem 822 - 12°
1001**Buenos Aires**
Tel: 31-6063,4,5,6 and 7
Telex: Public Booth N° 9
Cable: HEWPACK ARG

BOLIVIA
Stambuk & Mark (Bolivia) Ltda.
Av. Mariscal, Santa Cruz 1342
La Paz
Tel: 40626, 53163, 52421
Telex: 3560014
Cable: BUKMAR

BRAZIL
Hewlett-Packard do Brasil I.e.C. Ltda.
Avenida Rio Negro, 980
Alphaville
06400 Barueri **Sao Paulo**
Tel: 429-2148/9,429-2118/9

Hewlett-Packard do Brasil I.e.C. Ltda.
Rua Padre Chagas, 32
90000-**Pôrto Alegre**-RS
Tel: (0512) 22-2998, 22-5621
Cable: HEWPACK pôrto Alegre

Hewlett-Packard do Brasil I.E.C. Lttda.
Rua Siqueira Campos, 53
Copacabana
20000-**Rio de Janeiro**
Tel: 257-80-94-DDD (021)
Telex: 391-212-1905 HEWP-BR
Cable: HEWPACK Rio de Janeiro

CHILE
Calcagni y Metcalfe Ltda.
Alameda 580-Of. 807
Casilla 2118
Santiago, 1
Tel: 398613
Telex: 3520001 CALMET
Cable: CALMET Santiago

COLOMBIA
Instrumentación
Henrik A. Langebaek & Kier S.A.
Carrera 7 No. 48-75
Apartado Aéreo 6287
Bogotá, I.D.E.
Tel: 69-88-77
Cable: AARIS Bogotá
Telex: 044-400

COSTA RICA
Científica Costarricense S.A.
Calle Central, Avenidas 1 y 3
Apartado 10159
San José
Tel: 21-86-13
Cable: GALGUR San José

ECUADOR
Medical Only
A.F. Viscaíno Compañía Ltda.
Av. Rio Amazonas No. 239
P.O. Box 2925
Quito
Tel: 242-150,247-033/034
Cable: Astor Quito

Calculators Only
Computadoras y Equipos Electrónicos
P.O. Box 2695
990 Toledo (y Cordero)
Quito
Tel: 525-982
Telex: 02-2113 Sagita Ed
Cable: Sagita-Quito

EL SALVADOR
Instrumentacion y Procesamiento Electronico de el Salvador
Bulevar de los Heroes II-48
San Salvador
Tel: 252787

GUATEMALA
IPESA
Avenida La Reforma 3-48,
Zona 9
Guatemala City
Tel: 63627, 64786
Telex: 4192 Teletro Gu

MEXICO
Hewlett-Packard Mexicana, S.A. de C.V.
Torres Adalid No. 21, 11° Piso
Col. del Valle
Mexico 12, D.F.
Tel: (905) 543-42-32
Telex: 017-74-507

Hewlett-Packard Mexicana, S.A. de C.V.
Ave. Constitucion No. 2184
Monterrey, N.L.
Tel: 48-71-32, 48-71-84
Telex: 038-843

NICARAGUA
Roberto Terán G.
Apartado Postal 689
Edificio Terán
Managua
Tel: 25114, 23412,23454
Cable: ROTERAN Managua

PANAMA
Electrónico Balboa, S.A.
P.O. Box 4929
Calle Samuel Lewis
Cuidad de Panama
Tel: 64-2700
Telex: 3431103 Curunda, Canal Zone
Cable: ELECTRON Panama

PARAGUAY
Z.J. Melamed S.R.L.
División: Aparatos y Equipos Médicos
División: Aparatos y Equipos Científicos y de Investigación
P.O. Box 676
Chile-482, Edificio Victoria
Aaunción
Tel: 91-271, 91-272
Cable: RAMEL

PERU
Compañía Electro Médica S.A.
Los Flamencos 145
San Isidro Casilla 1030
Lima 1
Tel: 41-4325
Cable: ELMED Lima

PUERTO RICO
Hewlett-Packard Inter-Americas
Puerto Rico Branch Office
Calle 272
No. 203 Urb. Country Club
Carolina 00924
Tel: (809) 762-7255
Telex: 345 0514

URUGUAY
Pablo Ferrando S.A.
Comercial e Industrial
Avenida Italia 2877
Casilla de Correo 370
Montevideo
Tel: 40-3102
Cable: RADIUM Montevideo

VENEZUELA
Hewlett-Packard de Venezuela C.A.
P.O. Box 50933
Caracas 105
Los Ruices Norte
3a Transversal
Edificio Segre
Caracas 107
Tel: 35-00-11 (20 lines)
Telex: 25146 HEWPACK
Cable: HEWPACK Caracas

FOR AREAS NOT LISTED, CONTACT:
Hewlett-Packard
Inter-Americas
3200 Hillview Ave.
Palo Alto, California 94304
Tel: (415) 493-1501
TWX: 910-373-1260
Cable: HEWPACK Palo Alto
Telex: 034-8300, 034-8493

EUROPE, NORTH AFRICA AND MIDDLE EAST

AUSTRIA
Hewlett-Packard Ges.m b.H.
Handelskai 52
P.O. box 7
A-1205 **Vienna**
Tel: (0222) 351621 to 27
cable: HEWPAK Vienna
Telex: 75923 hewpak a

BELGIUM
Hewlett-Packard Benelux
S.A./N.V.
Avenue de Col-Vert, 1,
(Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494 paloben bru

CYPRUS
Kypronics
19, Gregorios & Xenopoulos Rd.
P.O. Box 1152
CY-**Nicosia**
Tel: 45628/29
Cable: KYPRONICS PANDEHIS
Telex: 3018

CZECHOSLOVAKIA
Vyvojova a Provozni Zakladna
Vyzkumnych Ustavu v Bechovicich
CSSR-25097
Bechovice u Prahy
Tel: 89 93 41
Telex: 121333

Institute of Medical Bionics
Vyskumny Ustav Lekarskej Bioniky
Jedlova 6
CS-88346 **Bratislava-Kramare**
Tel: 44-551-45-541

DDR
Entwicklungslabor der TU Dresden
Forschungsinstitut Meinsberg
DDR-7305
Waldheim/Meinsberg
Tel: 37 667
Telex: 112145

Export Contact AG Zuerich
Guenther Forgber
Schlegelstrasse 15
1040 **Berlin**
Tel: 42-74-12
Telex: 111889

DENMARK
Hewlett-packard A/S
Datavej 52
DK-3460 **Birkerod**
Tel: (02) 81 66 40
Cable: HEWPACK AS
Telex: 166 40 hpas

Hewlett-Packard A/S
Naverve j 1
DK-8600 **Silkeborg**
Tel: (06) 82 71 66
Telex: 166 40 hpas
Cable: HEWPACK AS

FINLAND
Hewlett-Packard OY
Nahkahousuntie 5
P.O. Box 6
SF-00211 **Helsinki** 21
Tel: (90) 6923031
Cable: HEWPACKOY Helsinki
Telex: 12-1563 HEWPA SF

FRANCE
Hewlett-Packard France
Quartier de Courtaboeuf
Boite Postale No. 6
F-91401 **Orsay** Cédex
Tel: (1) 907 78 25
Cable: HEWPACK Orsay
Telex: 600048

Hewlett-Packard France
Agency Régionale
"Le Saquin"
Chemin des Mouilles
B.P. 162
F-69130 **Ecully**
Tel: (78) 33 81 25,
Cable: HEWPACK Eculy
Telex: 31 06 17

Hewlett-Packard France
Agence Régionale
Péricentre de la Cépière
Chemin de la Cépière, 20
F-31300 **Toulouse-Le Mirail**
Tel.:(61) 40 11 12
Cable: HEWPACK 51957
Telex: 510957

Hewlett-Packard France
Agence Régionale
Aéroport principal de
Marseille-Marignane
F-13721**Marignane**
Tel: (91) 89 12 36
Cable: HEWPACK MARGN
Telex: 410770

Hewlett-Packard France
Agence Régionale
63, Avenue de Rochester
B.P. 1124
F-35014 **Rennes** Cédex
Tel: (99) 36 33 21
Cable: HEWPACK 74912
Telex: 740912

Hewlett-Packard France
Agence Régionale
74, Allée de la Robertsau
F-67000 **Strasbourg**
Tel: (88) 35 23 20/21
Telex: 890141
Cable: HEWPACK STRBG

Hewlett-Packard France
Agence Régionale
Centre Vauban
201, rue Colbert
Entrée A2
F-59000 **Lille**
Tel: (20) 51 44 14
Telex: 820744

Hewlett-Packard France
Centre d' Affaires Paris-Nord
Bâtiment Ampére
Rue de La Commune de Paris
B.P. 300
F-93153
Le Blanc Mesnil Cédex
Tel: (01) 931 88 50

GERMAN FEDERAL REPUBLIC
Hewlett-Packard GmbH
Vertriebszentrale Frankfurt
Bernerstrasse 117
Posttach 560 140
D-6000 **Frankfurt** 56
Tel: (0611) 50 04-1
Cable: HEWPACKSA Frankfurt
Telex: 04 13249 hpffmd

Hewlett-Packard GmbH
Technisches Buero Böblingen
Herrenbergerstrasse 110
D-7030 **Böblingen**, Württemberg
Tel: (07031) 667-1
Cable: HEPAK Böblingen
Telex: 07265739 bbn

Hewlett-Packard GmbH
Technisches Buero Dusseldorf
Emanuel-Leutze-Str. 1 (Seestern)
D-4000 **Dusseldorf** 11
Tel: (0211) 59711
Telex: 085/86 533 hpdd d

Hewlett-Packard GmbH
Technisches Buero Hamburg
Wendenstrasse 23
D-2000 **Hamburg** 1
Tel: (040) 24 13 93
Cable: HEWPACKSA Hamburg
Telex: 21 63 032 hphh d

Hewlett-Packard GmbH
Technisches Buero Hannover
Am Grossmarkt 6
D-3000 **Hannover-Kleefeld 91**
Tel: (0511) 46 60 01
Telex: 092 3259

Hewlett-Packard GmbH
Technisches Buero Nuremberg
Neumeyer Str. 90
D-8500 **Nuremberg**
Tel: (0911) 56 30 83/85
Telex: 0623 860

Hewlett-Packard GmbH
Technisches Buero München
Unterhachinger Strasse 28
ISAR Center
D-8012 **Ottobrunn**
Tel: (089) 601 30 61/7
Cable: HEWPACKSA München
Telex: 0524985

Hewlett-Packard GmbH
Technisches Buero Berlin
Keith Strasse 2-4
D-1000 **Berlin** 30
Tel: (030) 24 90 86
Telex: 18 3405 hpbln d

GREECE
Kostas Karayannis
18, Ermou Street
GR-**Athens** 126
Tel: 3237731
Cable: RAKAR Athens
Telex: 21 59 62 rkar gr

Analytical Only
"INTECO"
G. Papathanassiou & Co.
Marni 17
GR - **Athens** 103
Tel: 522 1915
Cable: INTEKNIKA Athens
Telex: 21 5329 INTE GR

Medical Only
Technomed Hellas Ltd
52 Skoufa Street
GR - **Athens** 135
Tel: 362 6972, 363 3830
Cable: etalak athens
Telex: 21-4693 ETAL GR

HUNGARY
MTA
Müszerügyi és Méréstechnikai
Szolgalata
Lenin Krt. 67
1391 **Budapest** VI
Tel: 42 03 38
Telex: 22 51 14

ICELAND
Medical Only
Elding Trading Company Inc.
Hafnarhvoli - Tryggvatotu
IS-**Reykjavik**
Tel: 1 58 20
Cable: ELDING Reykjavik

IRAN
Hewlett-Packard Iran Ltd.
No. 13, Fourteenth St
Miremad Avenue
P.O. Box 41-2419
IR-**Tehran**
Tel: 851082-7
Telex: 213405 HEWP IR

IRAQ
Hewlett-Packard Trading Co
Mansoor City
Baghdad
Tel: 5517827
Telex: 2455 Hepairaq ik
Cable: HEWPACDAD,
Baghdad Iraq

IRELAND
Hewlett-Packard Ltd.
King Street Lane
GB-**Winnersh**, Wokingham
Berks. RG11 5AR
Tel: (0734) 78 47 74
Telex: 847178/848179

ITALY
Hewlett-Packard Italiana S.p.A.
Via Amengo Vespucci 2
Casella postale 3645
I-20100 **Milano**
Tel: (2) 6251 (10 lines)
Cable: HEWPACKIT Milano
Telex: 32046

Hewlett-Packard Italiana S.p.A.
Via Pietro Maroncelli 40
(ang. Via Visentin)
I-35100 **Padova**
Tel: (49) 66 48 88
Telex: 41612 Hewpacki

Medical only
Hewlett-Packard Italiana S.p.A.
Via d'Agliardi, 7
I-56100 **Pisa**
Tel: (050) 2 32 04
Telex: 32046 via Milano

Hewlett-Packard Italiana S.p.A.
Via G. Armellini 10
I-00143 **Roma**
Tel: (06) 54 69 61
Telex: 61514
Cable: HEWPACKIT Roma

Hewlett-Packard Italiana S.p.A.
Corso Giovanni Lanza 94
I-10130 **Torino**
Tel: (011) 682245/659308

Medical/Calculators Only
Hewlett-Packard Italiana S.p.A.
Via Principe Nicola 43 G/C
I-95126 **Catania**
Tel.(095) 37 05 04

Hewlett-Packard Italiana S.p.A.
Via Amerigo Vespucci, 9
I-80142 **Napoli**
Tel: (081) 33 77 11

Hewlett-Packard Italiana S.p.A.
Via E. Masi, 9/B
I-40137 **Bologna**
Tel: (051) 30 78 87

KUWAIT
Al-Khaldiya Trading &
Contracting Co.
P.O. Box 830
Kuwait
Tel: 42 49 10
Cable: VISCOUNT

LUXEMBURG
Hewlett-Packard Benelux
S.A./N.V.
Avenue du Col-Vert, 1,
(Groenkraaglaan)
B-1170 **Brussels**
Tel: (02) 672 22 40
Cable: PALOBEN Brussels
Telex: 23 494

MOROCCO
Gerep
190, Blvd. Brahim Roudani
Casablanca
Tel: 25-16-76/25-90-99
Cable: Gerep-Casa
Telex: 23739

NETHERLANDS
Hewlett-Packard Benelux N.V.
Van Heuven Goedhartlaan 121
P.O. Box 667
NL-1134 **Amstelveen**
Tel: (020) 47 20 21
Cable: PALOBEN Amsterdam
Telex: 13 216 hepa nl

NORWAY
Hewlett-Packard Norge A/S
Nesveien 13
Box 149
N-1344 **Haslum**
Tel: (02) 53 83 60
Telex: 16621 hpnas n

POLAND
Biuro Informacji Technicznej
Hewlett-Packard
U1 Stawki 2, 6P
00-950**Warszawa**
Tel: 395962/395187
Telex: 81 24 53 hepa pl

UNIPAN
Zaklad Doswiadczalny
Budowy Aparatury Naukowej
U1 Krajowej Rady Narodowej 51/55
00-800 **Warszawa**
Tel: 36190
Telex: 81 46 48

Zaklady Naprawcze Sprzetu
Medycznego
Plac Komuny Paryskiej 6
90-007 **Lodz**
Tel: 334-41, 337-83

PORTUGAL
Telectra-Empresa Técnica de
Equipamentos Electricos S.a.r.l.
Rua Rodrigo da Fonseca 103
P.O. Box 2531
P-**Lisbon** 1
Tel: (19) 68 60 72
Cable: TELECTRA Lisbon
Telex: 12598

Medical only
Mundinter
Intercambio Mundial de Comércio
S.a.r.l.
Av. A.A de Aguiar 138
P.O. Box 2761
P - **Lisbon**
Tel: (19) 53 21 31-7
Cable: INTERCAMBIO Lisbon

RUMANIA
Hewlett-Packard Reprezentanta
Bd.N. Balcescu 16
Bucharest
Tel: 158023/138885
Telex: 10440

I.I.R.U.C.
Intreprinderea Pentru
Intretinerea
Si Repararea Utilajelor de Calcul
B-dul prof. Dimitrie Pompei 6
Bucharest-Sectorul 2
Tel: 12 64 30
Telex: 11716

SAUDI ARABIA
Modern Electronic Establishment
King Abdul Aziz str.(Head office)
P.O. Box 1228
Jeddah
Tel: 31173-332201
Cable: ELECTRA

P.O. Box 2728 (Service center)
Riyadh
Tel: 62596-66232
Cable: RAOUFCO

SPAIN
Hewlett-Packard Española, S.A.
Jerez No. 3
E-**Madrid** 16
Tel (1) 458 26 00 (10 lines)
Telex: 23515 hpe

Hewlett-Packard Española, S.A.
Milanesado 21-23
E-**Barcelona** 17
Tel: (3) 203 6200 (5 lines)
Telex: 52603 hpbe e

Hewlett-Packard Española, S.A.
Av. Ramón y Cajal 1
Edificio Sevilla, planta 9,
E-**Seville** 5
Tel: 64 44 54/58

Hewlett-Packard Española S.A.
Edificio Albia II 7 B
E-**Bilbao** 1
Tel: 23 83 06/23 82 06

Calculators Only
Hewlett-Packard Española S.A.
Gran Via Fernando El Catdlico, 67
E-**Valencia**-8
Tel: 326 67 28/326 85 55

SWEDEN
Hewlett-Packard Sverige AB
Enighetsvägen 1-3
Fack
S-161 20 **Bromma** 20
Tel: (08) 730 05 50
Cable: MEASUREMENTS
Stockholm
Telex: 10721

Hewlett-Packard Sverige AB
Ostra Vintergatan 22
S-702 40 **Orebro**
Tel: (019) 14 07 20

Hewlett-Packard Sverige AB
Frötallsgatan 30
S-421 32 **Vastra Frolunda**
Tel: (031) 49 09 50
Telex: 10721 via Bromma Office

SWITZERLAND
Hewlett-Packard (Schweiz) AG
Zürcherstrasse 20
P.O. Box 307
CH-8952 **Schlieren-Zurich**
Tel: (01) 730 52 40/730 18 21
Cable: HPAG CH
Telex: 53933 hpag ch

Hewlett-Packard (schweiz) AG
Château Bloc 19
CH-1219 **Le Lignon-Geneva**
Tel: (022) 96 03 22
Cable: HEWPACKAG Geneva
Telex: 27 333 hpag ch

SYRIA
Medical/Calculator only
Sawah & Co.
Place Azmé
B.P. 2308
SYR-**Damascus**
Tel: 16367, 19697, 14268
Cable: SAWAH, Damascus

TURKEY
Telekom Engineering Bureau
P.O. Box 437
Beyoglu
TR-**Istanbul**
Tel: 49 40 40
Cable: TELEMATION Istanbul
Telex: 23609

Medical only
E.M.A.
Muhendislik Kollektif Sirketi
Adakale Sokak 41/6
TR-**Ankara**
Tel: 175622

Analytical only
Yilmaz Ozyurek
Milli Mudafaa Cad No. 16/6
Kizilay
TR-**Ankara**
Tel: 25 03 09
Telex: 42576 Ozek tr

UNITED KINGDOM
Hewlett-Packard Ltd.
King Street Lane
GB-**Winnersh**, Wokingham
Berks. RG11 5AR
Tel: (0734) 78 47 74
Cable: Hewpie London
Telex: 847178/9

Hewlett-Packard Ltd.
Tralalger House,
Navagation Road
Altrincham
Cheshire WA14 1NU
Tel: (061) 928 6422
Cable: Hewpie Manchester
Telex: 668068

Hewlett-Packard Ltd.
Lygon Court
Hereward Rise
Oudley Road
Halesowen,
West Midlands B62 8SO
Tel: (021) 550 9911
Telex: 339105

Hewlett-Packard Ltd.
Wedge House
799, London Road
GB-**Thornton Heath**
Surrey CR4 6XL
Tel: (01) 684 0103/8
Telex: 946825

Hewlett-Packard Ltd.
c/o Makro
South Serviceholesale Centre
Wear industrial Estate
Washington
GB-**New Town**, County Durham
Tel: Washington 464001 ext. 57/58

Hewlett-Packard Ltd
10, Wesley St.
GB-**Castleford**
West Yorkshire WF10 1AE
Tel: (09775) 50402
Telex: 557355

Hewlett-Packard Ltd
1, Wallace Way
GB-**Hitchin**
Herts
Tel: (0462) 52824/56704
Telex: 825981

USSR
Hewlett-Packard
Representative Office USSR
Pokrovsky Boulevard 4/17-KW 12
Moscow 101000
Tel:294-2024
Telex: 7825 hewpak su

YUGOSLAVIA
Iskra-standard/Hewlett-Packard
Miklosiceva 38/VII
61000 **Ljubljana**
Tel: 31 58 79/32 16 74
Telex: 31583

SOCIALIST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard Ges.m.b.H.
P.O. Box 7
A-1205 **Vienna**, Austria
Tel: (0222) 35 16 21 to 27
Cable: HEWPAK Vienna
Telex: 75923 hewpak a

MEDITERRANEAN AND MIDDLE EAST COUNTRIES NOT SHOWN PLEASE CONTACT:
Hewlett-Packard S.A.
Mediterranean and Middle
East Operations
35, Kolokotroni Street
Platia Kefallariou
GR-Kifissia-**Athens**, Greece
Tel: 8080337/359/429
Telex: 21-6588
Cable: HEWPACKSA Athens

FOR OTHER AREAS NOT LISTED CONTACT
Hewlett-Packard S.A.
7, rue du Bois-du-Lan
P.O. Box
CH-1217 Meyrin 2 - **Geneva**
Switzerland
Tel: (022) 82 70 00
Cable: HEWPACKSA Geneva
Telex: 2 24 86

UNITED STATES

ALABAMA
8290 Whitesburg Dr., S.E.
P.O. Box 4207
Huntsville 35802
Tel: (205) 881-4591

Medical Only
228 W. Valley Ave.,
Room 220
Birmingham 35209
Tel: (205) 942-2081 2

ARIZONA
2336 E. Magnolia St.
Phoenix 85034
Tel: (602) 244-1361

2424 East Aragon Rd.
Tucson 85706
Tel: (602) 294-3148

***ARKANSAS**
Medical Service Only
P.O. Box 5646
Brady Station
Little Rock 72205
Tel: (501) 376-1844

CALIFORNIA
1430 East Orangethorpe Ave.
Fullerton 92631
Tel: (714) 870-1000

3939 Lankershim Boulevard
North Hollywood 91604
Tel: (213) 877-1282
TWX: 910-499-2671

6305 Arizona Place
Los Angeles 90045
Tel: (213) 649-2511
TWX: 910-328-6147

***Los Angeles**
Tel: (213) 776-7500

3003 Scott Boulevard
Santa Clara 95050
Tel: (408) 249-7000
TWX: 910-338-0518

***Ridgecrest**
Tel: (714) 446-6165

646 W. North Market Blvd
Sacramento 95834
Tel: (916) 929-7222

9606 Aero Drive
P.O. Box 23333
San Diego 92123
Tel: (714) 279-3200

COLORADO
5600 South Ulster Parkway
Englewood 80110
Tel: (303) 771-3455

CONNECTICUT
12 Lunar Drive
New Haven 06525
Tel: (203) 389-6551
TWX: 710-465-2029

FLORIDA
P.O. Box 24210
2806 W. Oakland Park Blvd.
Ft. Lauderdale 33311
Tel: (305) 731-2020

***Jacksonville**
Medical Service only
Tel: (904) 398-0663

P.O. box 13910
6177 Lake Ellenor Dr.
Orlando 32809
Tel: (305) 859-2900

P.O. Box 12826
Pensacola 32575
Tel: (904) 476-8422

GEORGIA
P.O. Box 105005
Atlanta 30348
Tel: (404) 955-1500
TWX 810-766-4890

Medical Service Only
***Augusta** 30903
Tel: (404) 736-0592

P.O. Box 2103
Warner Robins 31098
Tel: (912) 922-0449

HAWAII
2875 So. King Street
Honolulu 96814
Tel: (808) 955-4455
Telex: 723-705

ILLINOIS
5201 Tollview Dr.
Rolling meadows 60008
Tel: (312) 255-9800
TWX: 910-687-2260

INDIANA
7301 North Shadeland Ave.
Indianapolis46250
Tel: (317)842-1000
TWX: 810-260-1797

IOWA
1902 Broadway
Iowa City 52240
Tel: (319) 338-9466

KENTUCKY
Medical Only
Atkinson Square
3901 Atkinson Dr.,
Suite 207
Louisville 40218
Tel: (502) 456-1573

LOUISIANA
P.O. Box 840
3229-39 Williams Boulevard
Kenner 70063
Tel: (504) 443-6201

MARYLAND
6707 Whitestone Road
Baltimore 21207
Tel: (301) 944-5400
TWX: 710-862-9157

2 Choke Cherry Road
Rockville 20850
Tel: (301) 948-6370
TWX: 710-828-9684

MASSACHUSETTS
32 Hartwell Ave
Lexington 02173
Tel: (617) 861-8960
TWX: 710-326-6904

MICHIGAN
23855 Research Drive
Farmington Hills 48024
Tel: (313) 476-6400

MINNESOTA
2400 N. Prior Ave.
St. Paul 55113
Tel: (612) 636-0700

MISSISSIPPI
***Jackson**
Medical Service only
Tel: (601) 982-9363

MISSOURI
11131 Colorado Ave.
Kansas City 64137
Tel: (816) 763-8000
TWX: 910-771-2087

148 Weldon Parkway
Maryland Heights 63043
Tel: (314) 567-1455
TWX: 910-764-0833

NEBRASKA
Medical Only
7171 Mercy Road
Suite 110
Omaha 68106
Tel: (402) 392-0948

NEW JERSEY
W. 120 Century Rd.
Paramus 07652
Tel: (201) 265-5000
TWX: 710-990-4951

Crystal Brook Professional
Building
Eatontown 07724
Tel:(201) 542-1384

NEW MEXICO
P.O. Box 11634
Station E
11300 Lomas Blvd., N.E.
Albuquerque 87123
Tel: (505) 292-1330
TWX: 910-989-1185

156 Wyatt Drive
Las Cruces 88001
Tel: (505) 526-2484
TWX: 910-983-0550

NEW YORK
6 Automation Lane
Computer Park
Albany 12205
Tel: (518) 458-1550

201 South Avenue
Poughkeepsie 12601
Tel: (914) 454-7330
TWX: 510-253-5981

39 Saginaw Drive
Rochester 14623
Tel: (716) 473-9500
TWX: 510-253-5981

5858 East Molloy Road
Syracuse 13211
Tel: (315) 454-2486
TWX: 710-541-0482

1 Crossways Park West
Woodbury 11797
Tel: (516) 921-0300
TWX: 710-990-4951

NORTH CAROLINA
P.O. Box 5188
1923 North Main Street
High Point 27262
Tel: (919) 885-8101

OHIO
16500 Sprague Road
Cleveland 44130
Tel: (216) 243-7300
TWX: 810-423-9430

330 Progress Rd.
Dayton 45449
Tel: (513) 859-8202

1041 Kingsmill Parkway
Columbus 43229
Tel: (614) 436-1041

OKLAHOMA
P.O. Box 32008
Oklahoma City 73132
Tel: (405) 721-0200

OREGON
17890 SW Lower Boones
Ferry Road
Tualatin 97062
Tel: (503) 620-3350

PENNSYLVANIA
111 Zeta Drive
Pittsburgh 15238
Tel: (412) 782-0400

1021 8th Avenue
King of Prussia Industrial Park
King of Prussia 19406
Tel: (215) 265-7000
TWX: 510-660-2670

SOUTH CAROLINA
6941-0 N. Trenholm Road
Columbia 29260
Tel: (803) 782-6493

TENNESSEE
***Knoxville**
Medical Service only
Tel: (615) 523-5022

1473 Madison Avenue
Memphis 38104
Tel: (901) 274-7472

Nashville
Medical Service only
Tel: (615) 244-5448

TEXAS
P.O. Box 1270
201 E. Arapaho Rd.
Richardson 75080
Tel: (214) 231-6101

P.O. Box 27409
6300 Westpark Drive
Houston 77057
Tel: (713) 781-6000

205 Billy Mitchell Road
San Antonio 78226
Tel: (512) 434-8241

UTAH
2160 South 3270 West Street
Salt Lake City 84119
Tel: (801) 487-0715

VIRGINIA
P.O. Box 12778
No. 7 Koger Exec. Center
Suite 212
Norfolk 23502
Tel: (804) 461-4025/6

P.O. Box 9669
2914 Hungary Springs Road
Richmond 23228
Tel: (804) 285-3431

WASHINGTON
Bellefield Office Pk.
1203-114th Ave. S.E.
Bellevue 98004
Tel: (206) 454-3971
TWX: 910-443-2446

***WEST VIRGINIA**
Medical Analytical Only
Charleston
Tel: (304) 345-1640

WISCONSIN
9004 West Lincoln Ave
West Allis 53227
Tel: (414) 541-0550

FOR U.S. AREAS NOT LISTED:
Contact the regional office nearest you: Atlanta, Georgia; North Hollywood, California; Rockville, Maryland; Rolling Meadows, Illinois. Their complete addresses are listed above

***Service Only**

Part No. 30000-90008
Printed in U.S.A. 6/77

HEWLETT hp PACKARD

Sales and service from 172 offices in 65 countries.
5303 Stevens Creek Blvd., Santa Clara, California 95050