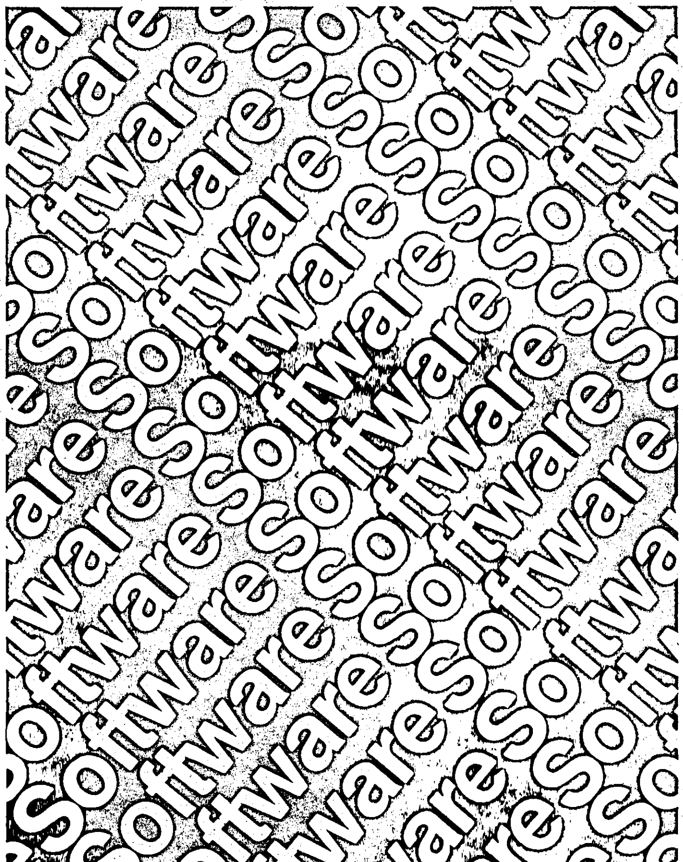
Honeywell



CP-6 CONCEPTS AND FACILITIES

CP-6

CONCEPTS & FACILITIES

SUBJECT

Defines the Projected Implementation of the Honeywell Control Program-Six (CP-6) Operating System

SOFTWARE SUPPORTED

Software Release B00 and beyond

ORDER NUMBER

CE26-01

September 1980

Honeywell

About This Manual

This general purpose manual introduces the reader with some technical awareness of software systems to the capabilities of the CP-6 system. This manual is intended for application and system programmers; transaction processing, word processing and I-D-S/II administrators; and system and EDP managers. The document includes both an overview of the CP-6 system and more detailed introductions to the CP-6 hardware configuration, software processors and operational modes.

Sections 1 through 6 of this manual contain an overview of the CP-6 hardware configuration, software processors and operational modes. Sections 7 through 18 examine the CP-6 system and operational modes in more detail. A set of appendices summarizing the major CP-6 utility processors, an appendix of CP-6 monitor services and a glossary of terminology are included at the end of the manual.

There is no prerequisite documentation to reading this manual. For the management-oriented reader interested in a limited overview of the CP-6 system, it is suggested that reading be limited to sections 1 through 6 and the first paragraph in each of the sections 7 through 18.

Section 1

Introducing the Honeywell CP-6 System

The Honeywell Control Program-Six (CP-6) System is a comprehensive, multi-use, distributed processing operating system designed to perform on Honeywell mainframes configured with Honeywell minicomputers.

The CP-6 system provides centralized services via five operational modes of access:

- Transaction processing
- Time sharing
- Batch processing
- Remote processing
- Distributed real-time processing

CP-6 access modes are supported with balanced service and no inherent emphasis on any single mode. Programs do not require alteration to run in any particular access mode.

These modes are designed to operate concurrently. Several programs utilizing different modes can be simultaneously resident in memory. The system design allows the user to select only the mode or modes required for a given task. The CP-6 system performs equally well whether a single mode or multiple modes are used. CP-6 functional elements are essentially the same for all programs regardless of the access mode.

CP-6 Transaction Processing allows multiprogramming depth for multiple queues, with processing divided into separately administered groups of terminals. Features include:

- Assurance of transaction completion.
- Administrative control over access and transaction prioritization.
- Larger amounts of processing time dedicated to each program in response to increased loads.
- Communication between transaction processing applications modules.
- Automatic journalization and recovery services.
- Advanced data base management provided through the CP-6 data management system, Integrated Data Store II (I-D-S/II).

CP-6 Time Sharing provides a highly productive environment designed to promote on-line program development and debugging. Features include:

- Up to 500 interactive terminals connected to the system.
- Rapid access to and response from the CP-6 system so that each time-sharing user appears to have the entire system dedicated to his task.

- Highly interactive response that is practically independent of system load.

- Access to all types of peripheral devices.

- Support of a wide variety of terminals.

- User definition of terminal characteristics (e.g., character code set, timing information, terminal features and cursor positioning) through the CP-6 terminal profile feature.

- Terminal access without translation, providing transparent control for special purpose devices.

CP-6 Batch Processing provides maximum utilization of system resources by minimizing conflicts in resource use. Features include:

- Concurrent processing of up to 500 batch jobs.

- Submission of batch jobs from on-line terminals or remote workstations.

- Channeling of batch jobs into the stream best able to handle the individual requirements of the job, consistent with throughput and resource constraints determined by installation management.

- Preservation of the batch queue during system recovery, and the recovery of jobs being processed at the time of system failure.

CP-6 Remote Processing provides flexible communication between the CP-6 system and a variety of remote terminals using synchronous protocols. Features include:

- Remote terminals and associated stations.

- HASP and 2780/3780 protocol support.

- Terminals can range from a simple card reader/line printer combination to a complete large-scale computer system with an assortment of peripheral devices.

- Time-sharing terminals, can be used as peripheral devices (line printers, etc.).

- Communication with any supported device at one or several remote sites.

- The ability to use any time-sharing terminal as the operator's console for a CP-6 workstation.

- The ability of a CP-6 system to act as a central site to several remote terminals and as a remote terminal to other computers simultaneously.

- Dynamic modification of terminal definition during system operation.

CP-6 Distributed Real-Time Processing allows the implementation of multiple sensor-based, real-time applications. Features include:

- Distribution of portions of real-time capability to real-time processors.

- Performance of data reduction and analysis by the host system.

- Performance of sensor-based applications on real-time processors, allowing a wide range of event-driven applications.

CP-6 SYSTEM FEATURES

The CP-6 system equals and exceeds the industry standards for performance, convenience, and cost-efficiency.

EASE OF USE

- A simple, yet comprehensive execution control language that is common to all access modes.
- An extensive HELP facility that provides information about the system and its processors.
- System default conventions that minimize the need for execution control commands.
- Terminal personality that includes type-ahead, echoplex, and a variety of escape and control keyins, providing an unparalleled interactive interface.
- Quick terminal response.
- Program and data file compatibility in all modes of access.
- A comprehensive remote batch system that allows entry of jobs from a variety of terminals.
- The ability to use installation-supplied command processors or data base managers to tailor system use to specific applications.

OPTIMIZED FILE MANAGEMENT

- A single central file management system.
- Files are compatible across operating modes and language processors.
- Device independent file access.
- Graduated levels of file access security.
- A comprehensive file backup system.
- File integrity assured by system recovery.
- Self-contained 'sets' of disk packs provided removable public file segments.
- Keyed, indexed, consecutive and relative file organizations.
- ANS standard tape management for both labels and standard blocking modes.

AN EFFICIENT MONITOR

- An event-driven, priority-adjustable scheduler.
- Full utilization of hardware addressing and security features.
- Shared re-entrant programs and system processors.

- Automatic sharing of user programs.
- A comprehensive, easily-accessed set of monitor services provided via a strong standard interface that ensures program independence from monitor version.
- High I/O performance via tree-structure file indexes with several forms of I/O caches and program-disassociated bufferings.
- Multiprogramming and multiprocessing.

MINIMAL OPERATIONS COST

- Small staff requirements for installation and system support.
- Easily maintained.
- System recovery which does not require operator intervention and automatically determines the appropriate level of recovery.
- System can be run without an operator in attendance.
- Availability of on-line hardware diagnostics at time-sharing terminals at both local and remote sites.
- Availability of on-line remote access software debugging and patching facilities.
- Full system use accounting.

SYSTEM HIGHLIGHTS

- An integrated performance monitor that measures system performance simultaneously with normal operation.
- Majority of operating system and processing code is written in a high-level structured language (PL-6).
- A modern, extensive data base management system that is interfaced with COBOL, APL, FORTRAN, PL-6, IDP, and assembly language.
- Communication with other operating systems through ANS labeled tape and the HASP and 2780 communications protocols.
- A sophisticated debugger that can be run in either the interactive or batch environment, and which possesses a comprehensive set of functions suitable for debugging FORTRAN, PL-6, COBOL, GMAP-6, and other language processors.
- Common calling sequences generated by all languages, allowing programs written in several languages to be loaded and run together.
- Up to 256K (one million bytes) of program procedure and data with up to 384K words (one and one-half million bytes) of additional data segments.
- Superior hardware, ensuring a secure environment.
- The ability to define many concurrent batch streams with priority, class, and dependent job scheduling.
- A common command language for both on-line and batch jobs.

- The ability to define any standard remote or local terminal as an operator console.
- APL, BASIC, COBOL, FORTRAN, and RPG II match or exceed current commercial state-of-the-art industry standards.
- Hierarchical budget accounting for control of system changes and usage.
- Remote communications concentrators provide fast local response, error control over long lines, and economical use of lines via full-duplex protocols.
- Communication groups (comgroups) provide communications between terminals and transaction processing user jobs, and between one or more user jobs.
- Comprehensive user documentation.

Section 2

The CP-6 System

This section contains a general overview of the CP-6 system, and references the section of this manual where each feature is described in more detail.

THE MONITOR

The CP-6 monitor functions as the major control element in the operating system. The monitor governs the order in which programs are executed and provides common services to all programs. The number and types of the programs in an operating system vary according to the user requirements at a particular installation. Each individual operating system consists of a selection of monitor routines and processing programs that are closely integrated for a given set of applications.

The monitor controls and schedules the use of the system resources including CPUs, main memory, secondary storage devices, spooled unit record devices, and terminals of all types. The monitor provides extensive services to the users, the system manager, the computer operator, and the hardware and software support engineers.

Because the monitor is central to the operation of the CP-6 system, references are made to its functions throughout this document.

THE HARDWARE

The CP-6 system is designed to run on Honeywell mainframes with minicomputers functioning as local real-time and communications processors. CP-6 hardware provides a flexible yet secure computing environment.

The CP-6 hardware is described in Section 3.

THE SOFTWARE PROCESSORS

The CP-6 system supports a set of software processors that satisfy a variety of computing requirements. These processors are categorized into four groups:

- Command and Control Processors
- Language Processors
- Utility Processors
- System Management Processors.

The CP-6 software processors are described in Section 4.

Each unit of work is packaged together as a 'job', regardless of the access mode used to enter the system. Jobs are also referred to as 'users', and are the CP-6 execution scheduling unit. (In most respects, a CP-6 user is equivalent to a "process" in other systems.) There are four types of CP-6 jobs: batch jobs, on-line jobs, transaction processing jobs, and ghost jobs.

With a batch job, the monitor knows the entire control stream and resource requirements before the job is put into execution. The monitor schedules batch jobs to optimize the use of resources. As a general rule, batch jobs are disconnected from human interaction as output is not delivered until the completion of the job. Errors or abnormal conditions occurring within a job cause the remainder of the job to be discarded unless the user program or job control commands initiate exceptional condition processing. Batch jobs can be submitted from a central site card reader, through an on-line terminal, or through a remote processing terminal.

An on-line job receives its control stream directly from the user at a time-sharing terminal. Resource requirements are not known to the monitor in advance, and are acquired as needed and when available. The user interactively handles unexpected occurrences. An on-line job can do everything a batch job can do, including executing cataloged procedures and accessing peripherals.

In a transaction processing job, each terminal interaction or transaction is formalized by the monitor in much the same way as a complete batch job step. This means that full system protection is provided for all elements of the transaction: input, output, and data base access. The installation may choose several levels of protection as required by the importance of each transaction.

Ghost jobs have a command stream, which is usually contained in a file, and can consist of multiple job steps. Ghost jobs are initiated at the request of the system, the operator (via a log-on process), or a privileged job (via a monitor service request).

All jobs, regardless of job type, enter the system through one or more modes of access. These modes are:

- CP-6 Transaction Processing, described in Section 11.
- CP-6 Time-Sharing, described in Section 12.
- CP-6 Batch Processing, described in Section 13.
- CP-6 Remote Processing, described in Section 14.
- CP-6 Distributed Real-Time Processing, described in Section 15.

SUPPORT SERVICES

A flexible system initialization procedure allows the system manager to define the system to reflect the hardware configuration, the number of users, the system features, and the processors to be included.

Operator communications inform the operator about set-up requirements, device errors that need attention, and the current batch queue. Users can send and receive operator messages. Several different interactive terminals can be used as operator consoles simultaneously, and different types of messages can be routed to the appropriate consoles. An operator's console can be used simultaneously as a time-sharing terminal.

On-line diagnostics and hardware exercisers are available to the support engineers.

Accounting information is maintained for users and several processors are provided for accessing this information. Interfaces are provided to allow the system manager to include his own accounting routines in the system.

Accounting may cover an entire job or an individual job step. Processors are available that allow the system manager to charge for the job. Rates can be changed dynamically and applied to a variety of classes of users.

Performance tuning parameters exist throughout the system and are used extensively in scheduling jobs. Most parameters can be modified dynamically by the system manager to tune the system to his requirements. Statistics gathering and analysis processors are included with the system as an aid in the tuning efforts.

CP-6 support services are described in Section 16 through 18.

Section 3

The CP-6 Hardware

CP-6 hardware provides many enhancements over previously available computer systems. The hardware features expanded virtual mapped memory and shared program facilities, and permits distribution of processing to multiple host and minicomputer systems. Hardware security features, fully utilized by the operating system, provide a secure, yet flexible, environment. The virtual mapping facilities and the security features are implemented in the I/O processors, resulting in faster throughput and reduced overhead.

THE CENTRAL PROCESSOR

The CP-6 system is designed to use Honeywell Series 60 and DPS 8 central processing mainframe systems, which provide extensive instruction sets, including packed decimal and floating point arithmetic, bit and byte string operations, and many powerful instruction addressing modes. Up to 16 million words (64 megabytes) of solid state memory can be supported on each host system. A combination of Input/Output Multiplexers (IOMs) and Micro-Programmed Controllers (MPCs) provide access to memory from peripheral devices. Up to four IOMs may be configured on a CP-6 system. Each IOM will support a number of MPCs.

Available host peripherals include:

- Card readers: 300/500/1050 CPM.
- Card punch: 100/400 CPM.
- Line printers: 1100/1600 LPM, 64 or 96 character set.
- Magnetic tape drives: dual density 9-track 800/1600 and 1600/6250 BPI, 75/125/200 IPS; and dual density 7-track 556/800 BPI (device support only - no managed tapes).
- Disk drives: Removable 402s and 451s, and nonremoveable 501s are supported, with the following available storage:

Disk	Available Storage
402 (100 MB)	88 million 8-bit bytes/disk
451 (200 MB)	177 million 8-bit bytes/disk
501N (600 MB)	618 million 8-bit bytes/head-disk assembly.

THE FRONT-END PROCESSOR

The CP-6 system uses Level 6 minicomputers as communication and real-time processors. Up to 12 minicomputers use firmware-driven microprocessors to achieve modularity with optimum configurability.

MULTIPROCESSING

The CP-6 multiprocessing facility enables from one to six central processors to be incorporated as part of the basic system in order to achieve greater processing power and/or greater reliability. All central processors have access to all physical memory and are capable of simultaneously executing programs residing in sharable domains. Only one copy of the operating system is employed to manage the central processor complex.

Only one processor (considered the primary) in the processor complex is permitted to perform system initiation and execute I/O response monitor procedures. The other processors (considered secondary) in the processor complex are prevented from executing seldom used but critical internal procedures, such as the monitor's allocation segment routines. Frequently used procedures, however, may be executed by any processor in the complex, and both primary and secondary processor procedures include file management (including I/O start) and CPU scheduling of users. Efficient processor gating techniques prevent the simultaneous access of critical system tables and the simultaneous execution of critical procedures.

CPUs may be dynamically reconfigured during system operation without shutting down the system. Secondary CPUs may be added or removed, and the CPU designated as primary may be changed via simple operator-initiated commands.

CONFIGURATION

Figure 3-1 shows a generalized CP-6 DPS configuration. In this figure, the components contained within the broken lines constitute the central system. For the most current information concerning available configurations, contact your local Honeywell CP-6 Marketing Representative.

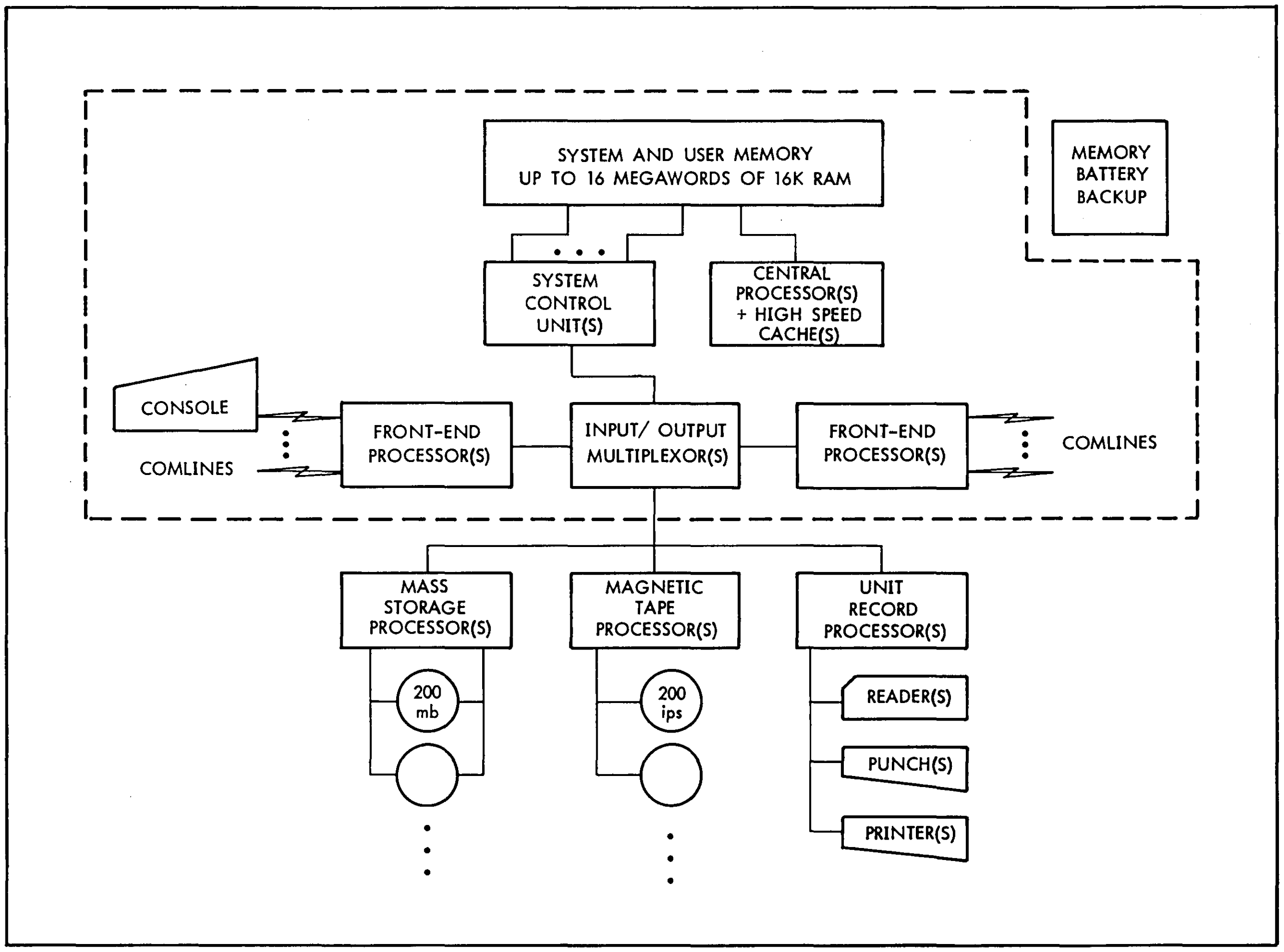


Figure 3-1. Generalized CP-6 DPS Configuration

Section 4

The CP-6 Software Processors

The CP-6 system supports a complete set of software processors that satisfies a variety of computing requirements:

- CP-6 command and control processors create an efficient user environment.
- CP-6 language processors feature compatibility with ANS standards.
- CP-6 utility processors offer a wide range of user services designed to increase programmer productivity.
- CP-6 system management processors help to reduce operations costs and promote efficient utilization of the entire CP-6 system.

Because the processors exist together in a self-consistent environment, the programmer is free to choose the right tool for the right job.

The CP-6 software processors are listed in Figure 4-1 in the order in which they are described in this section. In addition to these processors, user-developed processors and the following programs generally available in the CP-6 community are supported on the CP-6 system: GASP, IMSL, PASCAL, SLAM, SNOBOL, and SPSS.

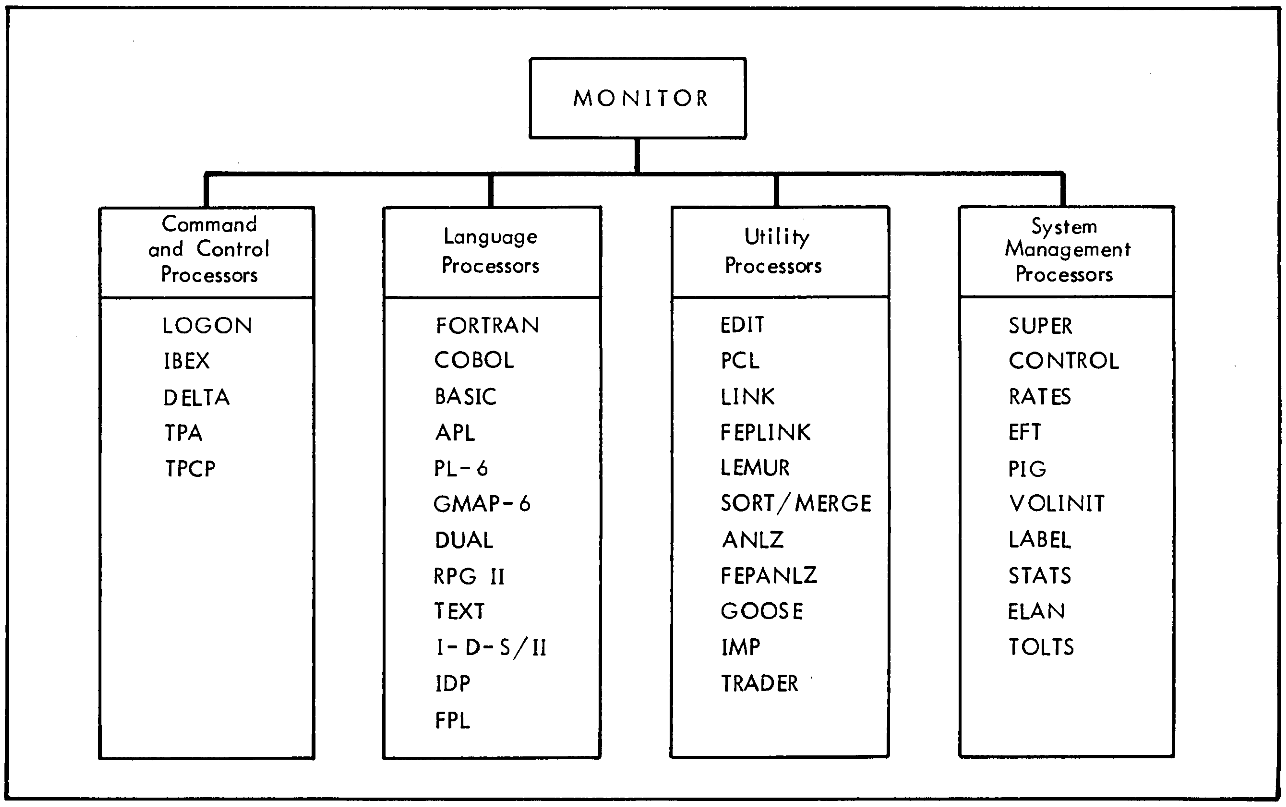


Figure 4-1. CP-6 Software Processors

COMMAND AND CONTROL PROCESSORS

LOGON

The LOGON processor controls access to the system by requiring the user to supply authorized identification information. Once this access is obtained, control is passed to IBEX.

IBEX

The Interactive and Batch Executive (IBEX) processor is the CP-6 execution control processor. The execution control commands that are interpreted by IBEX identify the user job, the tasks to be performed by the job,

and the resources required by the job. Other IBEX commands control interactive terminal operations. All batch jobs and interactive sessions require the use of execution control commands. The language is the same for both modes of processing (however, not all commands apply to both modes of processing). Appendix A is a summary of IBEX commands.

DELTA

The DELTA debugging processor is used to debug run units. The source code may have been written in any CP-6 assembler or high-level language. The language processors, in cooperation with the LINK loader, supply symbolic information to DELTA. The user describes the debugging requirements to DELTA in terms similar to the language in which the source program is written.

DELTA operates in both the batch and on-line modes. If the user is running in the time-sharing mode, conditions that occur in the user program are reported directly at the terminal. The user can then take immediate action to correct an error. In the batch mode, the user is restricted only to actions that can be preplanned.

DELTA allows the user to:

- Examine, insert, and modify program elements such as instructions, numeric values, and coded information (i.e., data in all its representations and formats).
- Control execution (including the insertion of break-points into a program) and requests for breaks on data changes within elements.
- Trace execution by displaying information at designated points in a program.
- Search programs and data for specific elements and subelements.

DELTA is designed and interfaced to the system in such a way that it may be called to aid debugging at any time, even after a program has been loaded and execution has begun. Appendix B is a summary of DELTA directives.

TPA

The Transaction Processing Administrator (TPA) is a privileged shared processor that is the Transaction Processing (TP) control processor. The TPA is responsible for monitoring the operation of an instance of TP. This processor is the administrative user of the TP comgroups (see Section 9), and receives all commands for the TP system. The TPA opens TP instance files and initializes instance tables. Thereafter, the processor checks the log-on id of each TP terminal, verifies passwords, keeps statistics, sends messages to the master control terminal or operator's console, and responds to commands from these devices.

TPCP

The Transaction Processing Command Processor (TPCP) is the shared command processor for TP applications. TPCP is associated with each Transaction Processing Application Program (TPAP) executing in an instance of TP. The TPCP invokes the TPAPs and supervises exit control for them. The actual processing of transactions and the creation of reports occurs in the TPAPs.

LANGUAGE PROCESSORS

FORTRAN

The CP-6 FORTRAN-77 compiler is compatible with essentially all the features of the American National Standards (ANS) FORTRAN 1978 X3.9, and includes extensions to that standard. Features of the CP-6 FORTRAN compiler include:

- CHARACTER variables.
- Addition of INCLUDE (system) capability.
- Line-by-line syntax-checking capability for time sharing.
- Expanded READ/WRITE capabilities.
- OPEN and CLOSE statements.
- I-D-S/II CALL interface.
- Conditional compilation capabilities.

COBOL

CP-6 COBOL offers a powerful and convenient programming language for implementation of business or commerical applications. COBOL is a standard compiler that conforms to American National Standards (ANS) COBOL X3.23-1974. I-D-S/II DML (Data Manipulation Language) capabilities are integral features of the compiler. The compiler accepts source program input from cards, remote terminals, user files, and the user copy library files. The compiler produces object-code compilation units from programs written in COBOL to form an executable run unit.

BASIC

CP-6 BASIC is a powerful compiler and programming language that is easy to teach, learn, and use, and is useful for a wide range of applications. CP-6 BASIC provides many significant enhancements over ANS minimal BASIC, including:

- A comprehensive set of statements, commands, and supplied functions; an extended MAT package, extensive character string manipulation facilities, and both ASCII and binary file I/O.
- The ability to share named data and data files between successively executed programs, and to access these files by direct statements.
- The ability to save and recall the complete working storage environment (including program, named data, and current status). This feature permits programs to be executed without forcing a recompile.
- The ability to carry out debugging operations at any time. The user can control BASIC's response to run-time errors.
- The ability to automatically trace program-flow and to specify breakpoints that interrupt execution, permitting immediate on-line debugging.
- Similarity in on-line and batch operations, which differ only in default device assignments and error response.
- The ability to execute most statements directly, allowing the on-line terminal to be used as a 'super' calculator.
- Conformity to CP-6 file conventions, allowing BASIC to access files created by other CP-6 processors and to create files that can be used by other CP-6 processors.
- The ability to seal programs, permitting the user to execute (but not modify, copy, or view) the programs and associated data.
- Structured programming statements.
- 31 character variable names.

APL

APL is a powerful and concise interpretive language, widely used by universities, engineers, and statisticians. APL also possesses features that make it particularly attractive for business applications, where user interaction and rapid feedback are key issues. CP-6 APL provides some special features, many of which are unavailable in other versions of APL:

- A compatible superset of IBM's APL/SV.
- On-line or batch operation.
- Increased I/O control facilities, providing easy access to standard CP-6 files.
- Shared access to files.
- Error and break control.
- Increased sub-string manipulation facilities.
- The ability to seal a workspace so that unauthorized users can load and execute, but not display or modify, the workspace contents.

- Accessibility by terminals without an APL character set.
- A fast formatter that facilitates report generation.
- An I-D-S/II interface.
- A comprehensive set of debugging aids.

PL-6

The PL-6 processor was designed specifically for the implementation of the CP-6 operating system. PL-6 combines the simplicity and directness of assembly language with the power and convenience of a higher-level language. Some of the outstanding features of this innovative and flexible language are:

- Provides a direct interface to the operating system.
- Uses a small run-time library.
- Produces a minimum of automatic storage.
- Provides handling of system-reported conditions via the ALTRETURN feature and asynchronous procedures.
- Adapts well to (but is not limited to) structured programming.
- Provides program control of register usage and calling sequences.
- Is geared for efficient use under the CP-6 system.

GMAP-6

GMAP-6 is a two-pass symbolic language assembler that translates symbolic machine language into relocatable binary machine instructions. GMAP-6 provides the convenience of a compiler with the flexibility of an assembler. GMAP-6 provides the programmer with a powerful programming tool that includes a complete set of pseudo-operations. GMAP-6 enables the programmer to use all machine instructions to design macros that provide convenient shorthand notations.

DUAL

The Dynamic Universal Assembly Language (DUAL) is an assembler that includes meta-language facilities. DUAL provides both system and application programmers with a powerful set of directives to reduce programming time, improve program checkout and develop languages to meet the needs of a specific application. In addition, through DUAL's special set of META directives, the programmer can extend DUAL so that it can translate a single phrase into a sequence of computer instructions.

renumbering of records, and context editing operations of matching, moving, and substituting for records selected by a range of line numbers and the presence or absence of specified character strings. File maintenance commands allow the user to build, copy, merge, and delete entire files. Appendix C is a summary of EDIT commands.

PCL

Peripheral Conversion Language (PCL) is a utility subsystem that provides information movement among card devices, line printers, tape devices, disk packs, terminals, and other peripherals. The flexible and powerful command language provides single and multiple file transfers with options for selecting, sequencing, formatting, encrypting, and converting data records. Additional file maintenance and utility commands are also included. Appendix D is a summary of PCL commands.

LINK

The LINK processor controls loading and linking of programs. LINK accepts object units (which are the output of compilers or assemblers) as input, resolves any linkages between them, and produces run units as its output. The processor may be directed to include object units from library files in the run unit and may also be directed to produce overlaid programs. LINK is available in both the batch and time-sharing modes.

FEPLINK

The FEPLINK processor controls loading and linking of all front-end resident programs. FEPLINK processing is performed in the host. FEPLINK accepts front-end object units (which are the output of front-end oriented processors such as FPL and DUAL) as input, resolves any linkage between them, and produces executables to be booted into the front-end processor as its output.

LEMUR

The LEMUR (Library Editor and Maintenance Utility Routines) processor builds library files from object files. LEMUR also edits existing library files and object files by performing insertion, deletion, and replacement of object units within these files. Library files built by LEMUR are accessed by LINK when constructing run files.

SORT/MERGE

SORT and MERGE are processors that provide a method of performing fundamental data manipulation processing:

1. Rearranging (sorting) records of multiple unordered files to a single specifically ordered file.

2. Combining (merging) records of multiple ordered files to a single ordered file.

SORT and MERGE may be run as a stand-alone processor, linked from a user program, or called directly from the system shared library.

ANLZ

The ANLZ processor allows the system programmer to analyze dumps. ANLZ displays relevant system information in an easily readable format.

FEPANLZ

The FEPANLZ processor performs boot and dump operations for front end processors, and is used to debug code that resides in the front end processors.

GOOSE

The GOOSE processor starts user-requested ghost jobs. A user with appropriate privileges can request that a ghost job be started immediately or can specify when a ghost job is to be started -- either at system startup or at a particular time of day.

IMP

IMP defines sequences and special characters that will be generated as a result of specified keystrokes at the terminal. These user-defined sequences or characters may be unique combinations of system escape sequences and special characters, or new special purpose functions suited to the individual user. IMP can be used to:

1. Redefine the keys on the keyboard of one terminal so that it looks like the keyboard of another terminal.

2. Define function keys to perform commonly used functions such as checking on jobs.

3. Define keys to generate often-used strings (such as lengthy variable names in a program).

TRADER

The Transaction to Application Definition Routine (TRADER), is a TP utility that defines the parameters necessary to associate transactions with Transaction Processing Application Programs (TPAPs) and Forms Programs (FPs). Information submitted to TRADER includes transaction names and types, TPAP and FP identification, and TPAP DCB assignments.

The TPCP command processor uses TRADER information to start TPAPs processing of transactions. The TPA uses TRADER information to invoke appropriate FPs to process transactions.

SYSTEM MANAGEMENT PROCESSORS

SUPER

The SUPER processor gives the system manager and authorized project managers control over access to the CP-6 system and the privileges extended to users. Through SUPER, the system manager and authorized project managers may add and delete users, specify how much main memory and disk storage space a user may have, specify how many central site magnetic tape units a user may use, grant certain users special privileges, (e.g., grant system programmers the privilege of examining, accessing, and changing the monitor), and individually authorize or deny access to the various processors for each user. SUPER is also the vehicle used to define the communications configuration of a CP-6 system.

CONTROL

The CONTROL processor allows the user to dynamically modify the system performance and control parameters.

RATES

The RATES processor allows the system manager to set relative charge weights on the utilization of system services and interactions.

EFT

The EFT processor allows the operations staff to back-up and restore files.

PIG

The Pack Set Initializer authorizes accounts on pack sets, mounts and dismounts pack sets, and reconstructs the available extent tables.

VOLINIT

The VOLINIT processor allows the operations staff to initialize CP-6 disk packs and to surface-check the devices to minimize errors.

LABEL

The LABEL processor writes ANS standard labels on tapes.

STATS

The STATS processor displays and collects performance data on a running system and produces snapshot files to be displayed later. Several forms of statistical summaries and history traces are available.

ELAN

The ELAN processor aids in the analysis of previously logged hardware and software failures.

TOLTS

The TOLTS processor allows the field engineers to run hardware tests and diagnostics on demand, while the remainder of the system continues operation. TOLTS is frequently used in conjunction with ELAN.

MONITOR SERVICES

The CP-6 monitor includes standard services that are available to user programs, regardless of the languages in which the programs are written. (Callable PL-6 routines may be required in certain cases.) Appendix E lists the CP-6 monitor services.

Section 5

CP-6 Documentation

Honeywell CP-6 documentation consists of two elements: the CP-6 manual set and CP-6 on-line documentation.

THE CP-6 MANUAL SET

The CP-6 manual library consists of a complete set of manuals that fully documents CP-6 software for the user. Figure 5-1 illustrates the CP-6 manual set, which is designed to meet the needs of different kinds of users working in each of the CP-6 environments.

The CP-6 manual set contains two types of manuals: references and guides. Designing the manual set to include these two types of manuals provides the user with both complete and tutorial information.

REFERENCES

A reference manual provides a detailed description of software. Concentrating on completeness and easy look up, a reference is organized in encyclopedic fashion -- usually alphabetically. As a result, it provides an in-depth description of a product for reference purposes, but is not intended as a stand-alone learning tool.

Reference manuals document each of the CP-6 environments. There are a total of 21 references in the CP-6 manual set. Within each environment, reference volumes are designed to both maximize document utility for the user and to minimize the number of manuals in the set. Where appropriate, utility is maximized by documenting processors in separate volumes. However, where functionality so indicates, documentation of processors is combined into a single volume to limit proliferation of manuals.

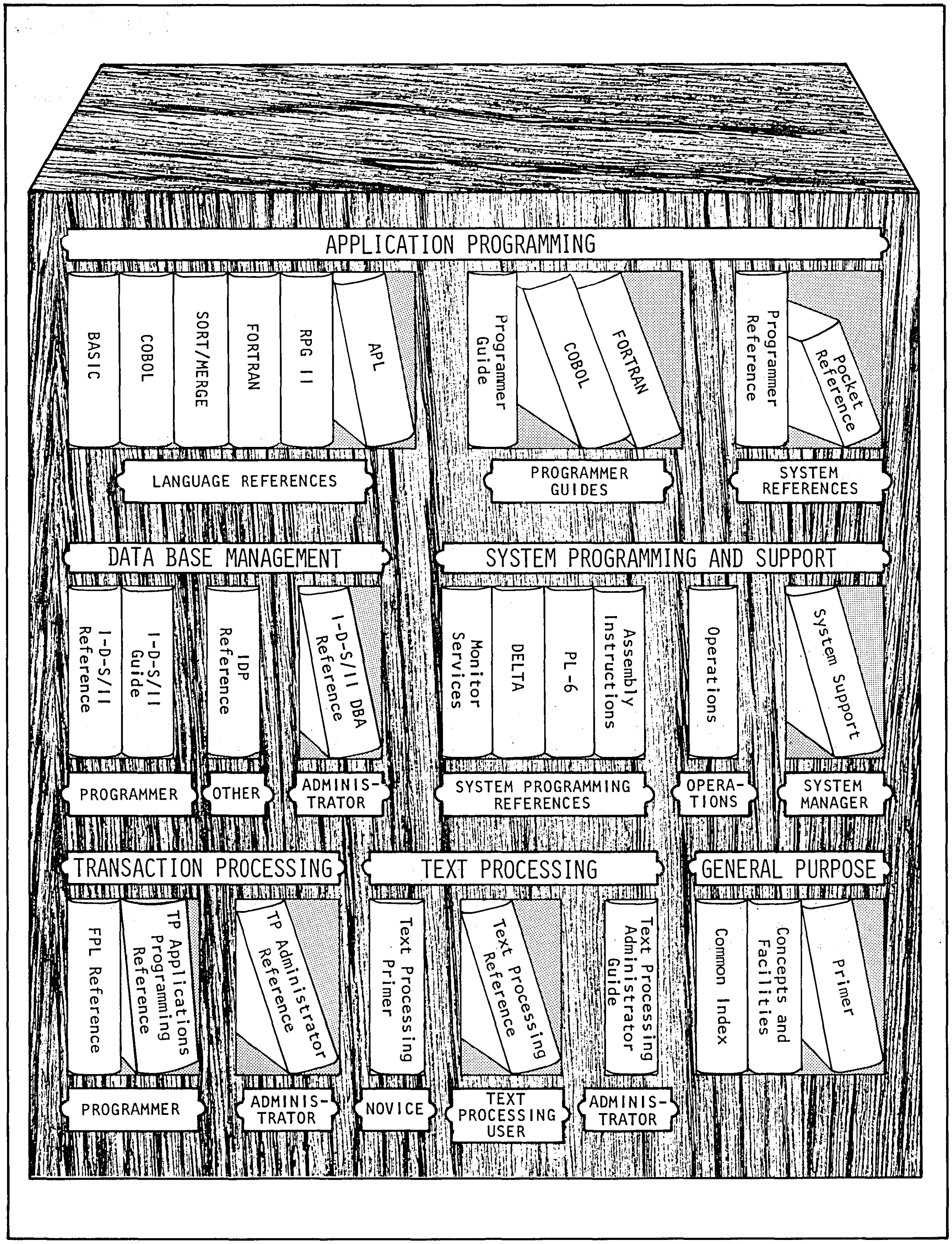


Figure 5-1. The CP-6 Documentation Set

For example, in the system programming and support environment, a single volume, the CP-6 System Support Reference, documents all the system processors of interest to the system manager. Likewise, in the application programming environment, a single volume, the CP-6 Programmer Reference, documents the several utility processors (EDIT, PCL, LINK and LEMUR) of interest to the application programming user. But, in the application programming environment, separate references document each of the language processors.

GUIDES

A guide provides tutorial information on a product. Concentrating on teaching how to use a product and ease of understanding, a guide is organized functionally, usually around examples. As a result, it serves as a textbook to be read as a learning document, but is not intended as a complete reference covering all aspects of a product.

The CP-6 manual set includes guides for APL, BASIC, COBOL, FORTRAN, and I-D-S/II programmers, as well as TP and Text Processing administrators.

Note that a subcategory of guides -- primers -- is included in the manual set. A primer introduces a non-sophisticated user to a CP-6 feature (or features). CP-6 primers introduce new, non-programming users to CP-6 (the CP-6 Primer) and to word processing (the CP-6 Text Processing Primer).

CATALOG OF DOCUMENTS

Table 5-1 is a list of the current CP-6 manual set divided into different user areas. Note that assignment of manuals to an area is not exclusive. Each user will tailor his or her library of documents to reflect his or her needs. Thus, a system programmer will make use of the CP-6 Programmer Reference, and an application programmer may need to refer to the CP-6 DELTA Reference.

This subsection contains descriptions of the manuals in the CP-6 user manual set.

Table 5-1. CP-6 Manual Set

User Area	Manual
Application Programming	Programmer Reference Pocket Reference COBOL Reference COBOL Programmer Guide FORTRAN Reference FORTRAN Programmer Guide SORT/MERGE Reference BASIC Reference APL Reference Programmer Guide RPG II Reference
Data Base Management	I-D-S/II Reference I-D-S/II Guide I-D-S/II Data Base Administrator Reference IDP Reference
System Programming and Support	Monitor Services Reference DELTA Reference PL-6 Reference Assembly Instruction Reference System Support Reference Operations Reference
Transaction Processing	FPL Reference TP Applications Programming Guide TP Administrator Guide
Text Processing	Text Processing Primer Text Processing Reference Text Processing Administrator Guide
General Purpose	Common Index Concepts and Facilities CP-6 Primer

APPLICATIONS PROGRAMMING MANUALS

CP-6 APL REFERENCE (CE38)   describes CP-6 APL language elements, statements, functions and system commands for application programmers already familiar with the language.

CP-6 BASIC REFERENCE (CE32)   describes the formats and uses of CP-6 BASIC commands, statements and functions for intermediate and advanced users of the language.

CP-6 PROGRAMMER GUIDE (CE55)   contains numerous examples that illustrate the environment of APL and BASIC programmers. Discussions are included on how to read and write data files, debug programs, interface with other language modules, use run time error control features, and use the CP-6 PCL and EDIT processors to aid in the support and maintenance of programs and working space files. In addition, this guide describes language extensions and enhancements (e.g., multi-line function definition capabilities).

CP-6 COBOL REFERENCE (CE29)   describes the formats, syntax rules, and general rules of CP-6 COBOL (an enhanced version of ANS COBOL X3.23-1974) for application programmers already familiar with the language. In addition to detailing the language, this manual contains information on compiling a COBOL program under the CP-6 operating system; using DELTA to debug a COBOL program; linking modules created outside COBOL; and interfacing COBOL with I-D-S/II.

CP-6 COBOL PROGRAMMER GUIDE (CE46)   contains a number of examples that illustrate the COBOL programmer's environment from creation of a source file through the development and production phases of a COBOL program. COBOL extensions and enhancements are described as well as newer COBOL ANS capabilities (e.g., table handling and the report writer feature). Discussions are included on debugging COBOL programs through DELTA, using compilation options, interfacing with other language modules, and using COBOL library facilities. In addition, this guide describes data formats and data manipulation, as well as interfaces between COBOL and CP-6 file management.

CP-6 FORTRAN REFERENCE (CE31)   describes the language elements, statements and features of CP-6 FORTRAN-77 (an enhanced version of ANS FORTRAN 1978 X3.9) for application programmers already familiar with the language. In addition to detailing the language, the manual contains information on compiling a FORTRAN program under the CP-6 operating system, and using DELTA to debug a FORTRAN program.

CP-6 FORTRAN PROGRAMMER GUIDE (CE47)   contains a number of examples that illustrate the FORTRAN programmer's environment from creation of a source file through the development and production phases of a FORTRAN program. In addition, FORTRAN extensions and enhancements are described as well as newer FORTRAN ANS capabilities (e.g., character data manipulation capabilities). Discussions are included on debugging FORTRAN programs through DELTA, using compilation options, interfacing with other language modules, and using run time error control features (IOSTAT and ERRSTAT).

CP-6 POCKET REFERENCE (CE42)   lists the syntax of the commands or directives of the following CP-6 utility processors: DELTA, EDIT, IBEX, LEMUR, PCL and SORT/MERGE. This pocket-sized book is intended as a quick reference to command and directive formats for application and system programmers. This command and directive summary includes a brief description of each of the CP-6 utility processors.

CP-6 PROGRAMMER REFERENCE (CE40)   describes the IBEX (Interactive and Batch Executive) commands used by application and system programmers to interface with the operating system. In addition to detailing IBEX commands, this manual includes an introduction to file and I/O management concepts, and details four CP-6 utility processors: EDIT, PCL, LINK and LEMUR.

CP-6 RPG II REFERENCE (CE37)   details the RPG II formats used to specify reports, describes the calculation operations available, and defines the procedures to compile and execute an RPG II program for readers who have some familiarity with programming and RPG. This manual includes examples and tutorial material to help new RPG users.

CP-6 SORT/MERGE REFERENCE (CE28)   describes for application and system programmers the SORT and MERGE processors and their directives. This manual includes examples and tutorial material to help new SORT/MERGE users.

DATA BASE MANAGEMENT MANUALS

CP-6 I-D-S/II REFERENCE (CE35) describes for the programmer the data manipulation language used to access an I-D-S/II data base application. This manual also summarizes the I-D-S/II subschema data description language and discusses CP-6 environment features, including I-D-S/II program execution, file assignments and access, journaling, recovery, and the subroutine library.

CP-6 I-D-S/II GUIDE (CE54) describes how to create an I-D-S/II data base. This manual helps the implementor of an I-D-S/II data base select the features that will best accomplish his goals and describes mechanical and operational aspects of defining, creating, loading, retrieving and maintaining data bases.

CP-6 I-D-S/II DATA BASE ADMINISTRATOR REFERENCE (CE36) describes the data base management environment from the perspective of those who control the design, creation, access controls and the use of a schema file, subschema file, and data base files. This manual describes the schema DDL, schema device media control DMCL, and subschema DDL used by the data base administrator. In addition to detailing the languages, the manual defines I-D-S/II data base concepts and design considerations, describes I-D-S/II data base privacy features, and details use of the DBUTIL processor to initialize, load, print and dump a data base.

CP-6 IDP REFERENCE (CE30) describes the CP-6 Interactive Data Base Processor (IDP) used to retrieve and display information contained in an I-D-S/II data base or on a data file. This manual is intended for anyone who wishes to use this processor to access data contained in a data base or on a data file. The manual describes the punctuation rules and syntax of IDP commands, and details the IDP Query language. In addition, the reference illustrates use of IDP and discusses retrieval strategy.

SYSTEM PROGRAMMING AND SUPPORT MANUALS

ASSEMBLY INSTRUCTIONS REFERENCE (DH03) defines the capabilities of GMAP-6, a set of machine instructions common to the Honeywell CP-6 and GCOS 8 operating systems used by system programmers. This manual describes the language's machine instructions, and discusses modes of operation, virtual memory addressing, and the representation of data.

CP-6 DELTA REFERENCE (CE39) describes for the programmer in any language the directives available to debug object code through the system's multilingual debug processor DELTA, and the debug facilities RUM and ANLZ. The parts of DELTA applicable to FORTRAN and COBOL are available in appendices to the respective reference manuals.

CP-6 MONITOR SERVICES REFERENCE (CE33) describes all the monitor services and some of the library services available with the CP-6 operating system used by programmers as well as those application programmers interested in accessing monitor services directly. The library services documented in this manual include the syntax parser, object unit generation services and source update services.

CP-6 OPERATIONS REFERENCE (CE34) describes system error codes and the system activities performed by the system operator to back up and recover files and to perform preventive maintenance. In addition, this manual describes operator communications (console attributes, usage and keyins), and the system processors used to manage packsets and tapes and to perform file backup and recovery: EFT, LABEL, PIG and VOLINIT.

CP-6 PL-6 REFERENCE (CE44) describes the PL-6 syntactic elements, statements and pre-processor facilities available to the system programmer interested in using this high level language in which the CP-6 operating system is programmed.

CP-6 SYSTEM SUPPORT REFERENCE (CE41) describes the processors used to perform system support activities at a customer's installation. This reference, intended for the system manager, details the following 11 system processors:

CONTROL	FEPCON	RATES	TIGR
EFT	GOOSE	STATS	VOLINIT
FEPANLZ	PIG	SUPER	

TRANSACTION PROCESSING MANUALS

CP-6 FPL REFERENCE (CE51) describes for the Forms programmer the Forms Processing Language used to build Forms Processing applications. This manual has separate sections for each of the divisions (i.e., identification, environment, data and procedure), each of which contains syntax rules and descriptions of statements and phrases.

CP-6 TP APPLICATIONS PROGRAMMING GUIDE (CE49) describes the TP environment and the interface between Transaction Processing Application Programs (TPAPs) and Forms Processing (FP) programs. This guide, directed to both the TPAPs and FP programmers, contains separate programming notes and examples for each kind of program, as well as discussions of data base and recovery considerations. The guide also contains a typical scenario and complete example of an instance of TP.

CP-6 TP ADMINISTRATOR GUIDE (CE50) describes the transaction processing environment from the perspective of the creator and monitor of an instance of TP. This guide details how to create and schedule an instance of TP, discusses TP accounting options and data, details TP and related security features, defines catastrophic recovery procedures and discusses performance criteria.

TEXT PROCESSING MANUALS

CP-6 TEXT PROCESSING PRIMER (CE53) introduces the non-programming reader to the basic features of CP-6 text processing. Through a series of examples, the reader learns how to log on and off the CP-6 operating system, initiate CP-6 word processing, and build, correct, format, and print a standard business letter.

CP-6 TEXT PROCESSING REFERENCE (CE48) describes the IMP, TEXT, EDIT, and PCL processors. This reference, which replaces the A00 release CP-6 TEXT Reference, is intended for the advanced text processing user, as well as the system analyst or text processing administrator who defines the text processing environment.

CP-6 TEXT PROCESSING ADMINISTRATOR GUIDE (CE52) describes the text processing environment from the perspective of those who set up one. This guide includes discussions of input terminal and output printer definition, effective use of the IMP processor, implementation of efficient file bulk storage, and selection of text processing features for various production tasks. In addition, the guide describes global file editing and other advanced applications.

CP-6 COMMON INDEX (CE43) collates all CP-6 manual indexes into a single volume. This manual contains a section-by-section description of each manual in the CP-6 manual set, as well as the collated master index of all CP-6 manuals. Note that the Assembly Instructions Reference (DH03) is not included.

CP-6 CONCEPTS AND FACILITIES (CE26) introduces the reader with some technical awareness of software systems to the capabilities of the CP-6 system. This manual is intended for application and system programmers; transaction processing, text processing and I-D-S/II administrators, and system and EDP managers. The document includes both an overview of the CP-6 system and more detailed introductions to the CP-6 hardware configuration, software processors and operational modes.

CP-6 PRIMER (CE45) introduces the non-programming user to CP-6 system features. Through a series of sample sessions, the reader learns how to perform common system operations, including logging on and off the CP-6 system; building and modifying files; listing and copying files to the terminal and the line printer; and building and executing a simple program.

CP-6 ON-LINE DOCUMENTATION

The Honeywell CP-6 system includes a form of on-line documentation that contains enough information to properly document an entire software product, yet is organized in a way that any desired item of information is easy to obtain.

The information in the CP-6 HELP facility is arranged in a tree-structure, based around a group of central messages that provide gateways into the various items of available documentation. When information about a specific item is requested (e.g., a command), only the briefest summary is printed (in this case the syntax of the command). Typing question marks causes successive layers to be printed containing parameter descriptions, conceptual descriptions, examples, related commands and concepts, down to the final level which points the user to the hard-copy manual where the feature is described in detail.

This querying process is not mandatory, as all layers can be displayed at once if so desired. Because all messages are structured in these layers, the end user obtains only the level of information that he or she requires and, in effect, constructs a manual that uniquely addresses his or her personal needs. For unsophisticated users, the system analyzes and identifies key-entry errors and then displays the correct format.

The on-line information is contained within a data base available throughout the system, capable of being utilized to produce other forms of documentation. Users can add messages to tailor the available information to their individual environment. By executing HELP requests as a file, interspersed with sample programs, a new form of documentation can be created where the information about a feature is printed and then verified by the actual execution of that feature.

CP-6 on-line documentation embraces the entire CP-6 system, dispensing reference and tutorial information according to the need of each individual user.

Section 6

CP-6 Programming Environment

CP-6 program development facilities are designed to promote the efficient creation of application programs through the use of a comprehensive set of utility and control processors.

OVERVIEW

A user creates, compiles, loads, and executes a program in the following manner:

1. The source language program is built as a file via the EDIT processor or is punched on cards.

2. The program is assembled or compiled by calling the appropriate processor. The command for calling this processor is the same in batch and time-sharing modes. The output of the assembly or compilation is a relocatable object unit, or, in languages such as BASIC or APL, a workspace.

3. The object unit or a set of associated object units is loaded by the LINK processor. The LINK processor combines object units into a single entity called a run unit. The user may specify that LINK is to produce an overlaid (tree-structured) program.

4. Program execution is initiated via an IBEX command (START), or by specifying (as a control command) the file identification of the file that contains the run unit.

5. Program debugging is aided by the DELTA processor, a multi-function debugger used to locate and correct program errors.

During these steps, the HELP facility can be used to obtain command syntax and other information about CP-6 and its processors.

SAMPLE CP-6 SESSION

Figure 6-1 shows a sample CP-6 session. The left hand pages show annotations that match the letters on the right or sample program.

Although Figure 6-1 does not illustrate all the CP-6 program development facilities, it does build, compile, link, execute and debug a program.

Figure 6-1: Annotation

A. The user initiates communication by connecting the terminal to the CP-6 system. If the user's terminal is hardwired, the connection is made by turning on the terminal. If the user's terminal is linked via an interface, the user follows local dial-up procedures. When the user makes connection with the CP-6 system, the system requests that the user enter a recognition character. The user enters the recognition character which is not echoed (displayed) on the terminal.

B. The system identifies itself in a way that is standard for the installation. Identification information will normally include time and date of connection, as well as logical and physical connection information.

C. system requests that the user log on and the user enter the account, name and password. As a security aid, this log-on information is not echoed on the terminal.

D. The system confirms the log on. This installation standard message will normally include time and date of the log on.

E. The IBEX command processor prompts the user to enter an IBEX command with the exclamation point character. The user enters the TERMINAL command to request terminal status information, which is output to the user's terminal.

F. IBEX prompts for another command, and the user invokes the EDIT processor used to build and manipulate source language and data files. The EDIT processor acknowledges that it has been invoked.

G. The EDIT processor prompts the user to enter an EDIT command with an asterisk. The user enters a command to set FORTRAN format tab stops.

H. The EDIT processor prompts the user for another command, and the user opens a new file and assigns it the name SITRI.

I. The EDIT processor prompts the user to enter successive file lines (also called records) with line numbers (also called record keys). In response to the first line number, the user enters the first line of a FORTRAN program. This process continues until the user responds to a line number prompt with an immeditate RETURN (see line 11). In this example, the user creates a program to read three variables, calculate their square roots, add the resultant values and write out all calculated values. As it appears at this point, the source program contains errors so that data manipulation features of the EDIT processor can be demonstrated.

J. The EDIT processor prompts for another EDIT command, and the user enters three chained commands (linked together with the semi-colon character) that:

   - Identify a search record (SE6)
   - Replace the first 2 blank characters in the record with the value 50 (/ /S/50/)
   - Request that the corrected line be echoed (TX).

   The EDIT processor types the manipulated record as corrected.

```
A { ?please type a left parenthesis

B {  *** CP-6 AT YOUR SERVICE, LADC L66A
  { 13:44 08/14/80 FEP #0001 PATH#000B LINE#1700

C { LOGON PLEASE:UACCT,UNAME,PSWRD

D {     *** SYSID# 126425 ON LADC L66A AT 12:44:09.57 AUG 14 '80


  { !TERMINAL
  {    TERMINAL ATTRIBUTES:
E {       NODE-PORT = 1-1700  LINE SPEED = 1200  PROFILE NAME = XRX850
  {       ON: TAB SIMULATION,RELATIVE TABBING,SPACE INSERTION,DISPLAY INPUT,
  {       APL LOWER CASE,SCROLL,PRINT HALT,RELATIVE PAGE,SAVED INPUT SIZE=3
F { !EDIT
  { EDIT HERE
G { *TA F
H   *BUILD SITRI
  {     1.000         WRITE (6,100)
  {     2.000 10      READ (5,200) X,Y,Z
  {     3.000         IF (X) 20,50,20
  {     4.000 20      D = SQRT(X**2+Y**2+Z**2)
  {     5.000         WRITE (6,300) X,Y,Z,D
I {     6.000         STOP
  {     7.000 100     FORMAT(7X,1HX,11X,1HY,11X,1HZ,11X,1HD)
  {     8.000 200     FORMAT (3E)
  {     9.000 300     FORMAT (4(1X,E11.3))
  {    10.000         END
  {    11.000
J   *SE6;/  /S/50/;TX
        6.000 50      STOP
```

Figure 6-1. Sample CP-6 Program

Figure 6-1: Annotation (cont)

K. The EDIT processor prompts for a command. The user enters an unrecognized command. The EDIT processor responds with **Eh?, and prompts for another command.

L. The user responds to the command prompt by entering a ? to obtain further information about the error condition that caused the EDIT processor to type **Eh?

M. The EDIT processor prompts for a command, and the user invokes the HELP processor again to examine the syntax of the EDIT command IN. The HELP processor displays the requested information.

N. The EDIT processor prompts for a command, and the user invokes the HELP processor for the next level of HELP information -- parameter descriptions. The HELP processor displays the requested information.

O. The EDIT processor prompts for a command and the user responds by entering an IN command that is syntactically correct. The EDIT processor prompts for the insert line by typing the line number, and the user enters the record data (note that a table precedes the data).

P. The user responds to the next EDIT prompt by entering the END command to return control to IBEX.

Q. IBEX prompts for a command and the user invokes the EDIT processor to open another new file. This time, the user builds a 3-record data file called CHANT.

R. The user responds to the next IBEX prompt by invoking the FORTRAN processor to compile the source program in file SITRI and write the object code to file DAEMON. (If file DAEMON already exists, the file will not be replaced; instead, a diagnostic message will be produced.)

S. The FORTRAN compilation results are displayed at the terminal, indicating an error-free compilation.

T. The user responds to the next IBEX prompt by opening another new file named GOETIA. The user build an execute (XEQ) file; that is, a file of IBEX commands. This file will subsequently be submitted as a job file for execution. When executed, the commands (records) in this file will:

- Establish the file CHANT as the source of input.
- Establish the user's terminal (ME) as the output destination for results of execution.
- Link and execute the object code contained in file DAEMON.

U. The user responds to the next IBEX prompt by executing the XEQ file.

V. The system displays the results of execution as follows:

- Each IBEX command in the XEQ file is echoed as it is executed.
- The results of the link and execute process are displayed on the terminal as a result of establishing the user's terminal as the output device.

```
K  { *IN LINE 5.5
     ** Eh?
L  { *?
     ** A syntax error was detected at 4
     *HELP IN
M  {
      Syntax:    IN[n][,i]
     *?
N  {
       n = key of first insertion. Default=highest key in record range + i.
       i = increment for following insertions. Default=1 or last-specified i.
O  { *IN 5.5
         5.500        GO TO 10
P  { *END
     !BUILD CHANT
Q  {       EDIT HERE
         1.000 1.0,2.0,3.0
         2.000 1.0,1.0,1.0
         3.000 0.0
         4.000
R  { !FORTRAN SITRI OVER DAEMON
     FORTRAN 77 VERSION A03  AUG 14 '80

     FORTRAN 77 VERSION A03 SOURCE=SITRI     COMPILE UNIT 001 AUG 14 '80 13:53:56.51       6

S  { *    1.000>     1:       WRITE (6,100)
        10.000>     11:       END
      ERRORS FOUND: 0             TOTAL ERRORS FOUND: 0
      ERR SEVERITY LEVEL: 0       MAX SEVERITY LEVEL: 0

     !BUILD GOETIA
     EDIT HERE
T  {     1.000 !SET F$5 CHANT,FUN=IN
         2.000 !SET F$6 ME
         3.000 !RUN DAEMON
         4.000
U  { !XEQ GOETIA
     *  :SHARED_COMMON_A01.:SYS (SHARED LIBRARY) ASSOCIATED.
             * * ALLOCATION SUMMARY * *
      PROTECTION            LOCATION       PAGES

      DATA                         0           1
      PROCEDURE                 2000           1
V  {  READ ONLY                    0           1
     *  NO LINKING ERRORS.

            X           Y           Z           D
        .100E+01    .200E+01    .300E+01    .374E+01
        .100E+01    .100E+01    .100E+01    .173E+01
      *STOP*
```

Figure 6-1. CP-6 Sample Program (cont)

Figure 6-1: Annotation (cont)

W. The user responds to the next IBEX prompt by requesting a listing of the files existing in the log on account. The four files created during this session are listed as the only four files in the account.

X. The user responds to the next IBEX prompt by directing that a listing of the source file be created for printing at the output destination PR@DOCUMENT. The device identification is known to the system.

Y. Another COPY command directs that a listing of the object code be created for printing at another output destination, LP@UPSTAIRS.

Z. The user responds to the next IBEX prompt by requesting that all queued output files be printed.

AA. The user responds to the next IBEX prompt by submitting the XEQ file for execution as a batch job. Note that no JOB command is required. The system responds by displaying the job identification of the submitted job.

BB. The user responds to the next IBEX prompt by invoking the BASIC processor. The BASIC processor acknowledges that it has been invoked.

CC. The BASIC processor prompts for input with the > character. The user responds by :

- Invoking AUTO mode.
- Entering statements that will find the square roots of three variables, and printing calculation results.
- Establishing the data file CHANT as an input file.
- Executing the contents of the work area.
- Sealing and saving the work area.
- Terminating use of the BASIC processor.

DD. The user responds to the next IBEX prompt by logging off the system.

EE. The system responds to the log off by printing terminating usage and accounting information.

NOTE: This figure is a file image copy of an actual CP-6 programming session.

```
W  { !L
   {   CHANT         DAEMON        GOETIA        SITRI
X  { !COPY SITRI TO PR@DOCUMENT
   {  ..COPYing
Y  { !COPY DAEMON TO LP@UPSTAIRS
   {  ..COPYing
Z  { !PRINT
AA { !BATCH GOETIA
   { Job 58825 Submitted
BB { !BASIC
   { BASIC A01 HERE

   { >AUTO
   { 10 DEF FND(X,Y,Z)=SQR((X**2)+(Y**2)+(Z**2))
   { 20 PRINT " X"," Y"," Z"," D"
   { 30 FOR A = 1 TO 2
   { 40 INPUT #1;A,X,Y,Z
   { 50 PRINT X,Y,Z,FND(X,Y,Z)
   { 60 NEXT A
   { 70 STOP
   { 80 END
CC { 90
   { >OPEN "CHANT" TO 1,INPUT
   { >RUN
   {  X             Y             Z             D
   {  1             2             3             3.74166
   {  1             1             1             1.73205
   {
   { HALT AT LINE 70
   { >SEAL LEMEGETON
   { LEMEGETON SAVED AND SEALED
   { >SYS
DD { !OFF
EE { CON=00:00:14:23 EX=00:00:03.97 SRV=00:00:10.10 PMME=  1238 CHG=    5.76
```

Figure 6-1. CP-6 Sample Program (cont)

BUILDING A PROGRAM

EDIT is a context edit for the creation, modification, and manipulation of textual files. EDIT is desdigned for on-line use; batch users without access to a time-sharing terminal generally punch programs on to cards.

All EDIT data is stored on disk in keyed files of variable length records. Through EDIT, a user can:

- Create a sequenced text file.
- Insert, delete, reorder and replace lines or groups of records within a file.
- Print and renumber file lines selectively.
- Merge part of one file into another.
- Select records for intra-record editing based on the presence or absence of specified character strings.
- Perform context editing operations that delete, move, and substitute character strings within a previously selected set of records.
- Maintain files. The user can build, copy and delete whole files of text lines.

Appendix C is a summary of EDIT commands.

COMPILING A PROGRAM

IBEX (Interactive and Batch Executive) is the CP-6 command processor. IBEX is an interface between the user and the operating system. IBEX interprets the CP-6 execution control language, the repetoire of IBEX commands. These commands control the construction and execution of programs and provide communication between a program and its environment. CP-6 processors are called by specifying the processor name in an IBEX control command. All jobs require the use of execution control language.

Appendix A is a summary of IBEX control commands.

LINKING AND EXECUTING A PROGRAM

The LINK processor controls the linking of programs. LINK accepts as input object units (which are the output of compilers or assemblers), resolves any linkage, and produces run units as output.

An overlaid program is a tree-structured program that has only one node (the root node) resident in main memory for the duration of the program execution. The other nodes are called for by a resident node and brought in as needed. They may reside (at different times) in the same main memory area, thus reducing the amount of main storage required to contain the entire program.

If the program is to be overlaid, the overlay specification is given to the LINK processor in the LINK command. It is the user's responsibility to plan the relationship of the nodes within the program.

The example in Figure 6-2 consists of four paths, any one of which may be present in main storage at any given time. Node A is the root of the program and is never overlaid by another node. Any path may be loaded into main storage and overlaid as many times as required by the program. All nodes of the run unit file are saved in disk storage. When a node that has been overlaid is called again by the executing program, the original copy is loaded from the disk. Therefore, any communication between two overlay segments (e.g., D and E in Figure 6-2) must be done in a part of the backward path common to both. Although Figure 6-2 illustrates a single tree structure, most actual overlaid programs consist of two parallel trees: one for data and one for programs. Both are fetched by a single node load call.

The user can direct IBEX to initiate execution of user programs that are in run unit form in two ways:

1. Simply by specifying the file identification of the file that contains the run unit.

2. Through the START command.

DEBUGGING A PROGRAM

DELTA is a universal debugger used for non-interpretive languages. Each language processor produces a debug schema defining the location and characteristics of each symbol defined within a program. This schema allows the programmer to communicate with DELTA using the exact symbols that appear on the program listing or the source program line numbers.

DELTA operates in both the batch and on-line access modes. In time-sharing, conditions in the user program are reported directly to the user's terminal. The user then interacts with DELTA to correct errors. In batch, the user must pre-plan the debugging session, entering the desired commands to DELTA within the IBEX command stream; the user is restricted only in that immediate interaction with DELTA is not possible.

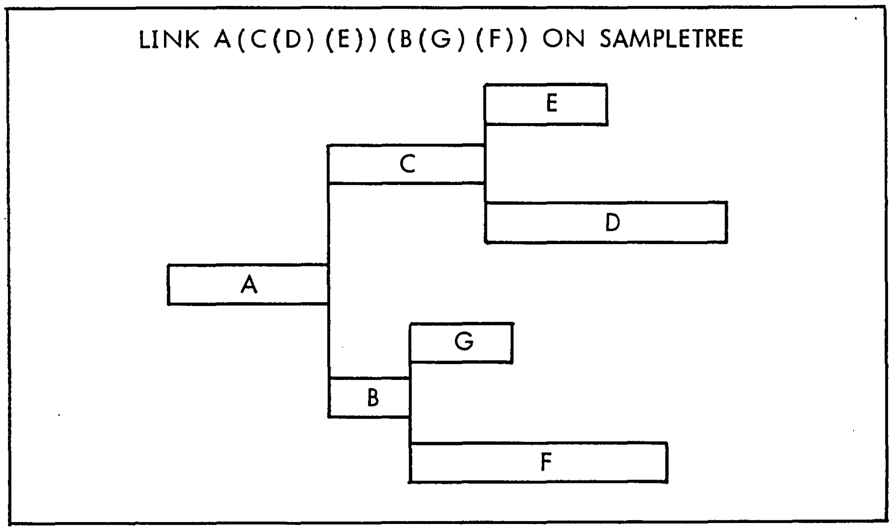


Figure 6-2. Sample Tree Structure

Tracing ability is provided at several levels: on all entry points, on a specific entry point, or on any condition which causes a break in the sequence of instruction execution. Additionally, a history mode can be set to save trace information for later examination. The DELTA user can:

- Interrupt the flow of program execution according to specified conditions.
- Examine, insert, and modify program elements such as instructions and data.
- Trace the flow of program execution.
- Obtain snapshots of post-mortem dumps.

The user directs DELTA activities through directives. DELTA directives can be catagorized into three groups:

1. Processor Directives that set parameters for the debugging session and cause DELTA to perform functions not related to the program itself.
2. Execution Control Directives that set procedure, data, and event breakpoints.
3. Memory Display and Modification Directives that print and/or modify memory.

These directives are considered direct, stored, or attachable depending on the time of activation. A direct directive is activated immediately. A stored directive is activated when a specified condition occurs. An attachable directive can either be invoked directly or as part of a stored command.

Appendix B is a summary of DELTA directives.

Section 7

CP-6 File and Device Management

CP-6 file and device management provides significant data organizing and input/output services. CP-6 I/O is device independent, allowing programs to be written without specific knowledge of the file or device at which I/O will actually take place. Default device assignments make specific selection necessary only when the programmer wishes to perform an unusual task. Pack sets and the EFT (Efficient File Transfer) processor provide protection from file destruction in the event of disaster. In addition, support of ANS tape formats allows the programmer to transfer information to and from other computer systems.

ORGANIZATION AND ACCESS METHODS

A file is an organized collection of information. This collection of information may consist of one or more programs, one or more sets of data, or some combination of programs and data. Under the CP-6 system, a user always accesses files through the monitor -- never directly. An option does exist, however, that allows a user to deal with a non-standard set of data on an unlabeled magnetic tape as though it is being accessed directly.

The monitor maintains an Account Directory which consists of account numbers, and for each account number, the address of a directory of files (termed a File Directory) for that account. A File Directory consists of file names and, for each file name, the address of a table containing file attributes and that file's location. The File Directory also contains access information for the account.

The File Information Table (FIT) contains additional information on each file. This information indicates who may access the file and how the file may be accessed, and can also include a list of which accounts may perform certain privileged operations. To access a file, a user must be running under an account that is authorized to access both the file's account and the file; the user must provide the proper password (if one exists). The FIT also contains various file-associated dates, special installation information, and the file type.

FILE FUNCTION AND DISPOSITION

A file can be opened for one of three functions: creation, input, and update. There are two possible dispositions for a file -- either to save it or to release it. Whether to save or release a file may be specified either when the file is opened or when the file is closed. If a file is opened in the save mode, the user can close the file in the release mode. However, a file opened in the release modes cannot be closed in the save mode. If the disposition of a file is not specified, the default (save or release) depends on whether or not the file is cataloged.

A user can request the opening of an existing file, the replacement of a file with a file of the same name, an error return to the user indicating that a file of the same name already exists, or the creation of a new file if no file of the same name exists.

FILE ORGANIZATION

The information in a file is structured in one of six ways: keyed, indexed, consecutive, relative, random, or unit-record. The type of structure is called the organization of the file and is a file attribute. The information in a file may be accessed in one of two ways: directly (by supplying the unique identifier of the record) or sequentially (by using the ordering relationship of the records).

KEYED FILES

In a keyed file, each record has an identifying key associated with it. A key consists of a byte string, with the first byte stating the number of bytes in the string. The contents of each byte may be a binary number or the binary representation of some character. A key can consists of up to 255 bytes.

As the file is being created, a master index is also created with an entry for each keyed record in the file. The keys are stored in collating sequence so that the file can be accessed sequentially. The entry contains such information as the key, the file-relative disk address of the record, the size of the record, and the position of the record within the blocking buffer.

Keyed files have a multilevel index structure to provide fast direct access. The higher level index block entry points to index blocks at the next lower level (i.e., one entry per block) and the entries in the lowest level (called level 0) point to data records.

The user has control over when and if the higher-level structure should be rebuilt. The system, however, automatically manages the structure without explicit user intervention.

INDEXED FILES

The indexed organization is implemented as a special case of the keyed file organization. The keys must be fixed length, less than 256 characters, and contained as a contiguous field within the data record.

CONSECUTIVE FILES

Consecutive files contain records that are organized in a consecutive manner, i.e., there are no identifying keys. The records can only be accessed sequentially.

The principal benefit of using consecutive files is a reduction in the amount of disk space required for the files, and a consequent reduction in time required to traverse the files.

RELATIVE FILES

A relative file consists of fixed-length records in a fixed-length file which is pre-allocated and effectively initialized at creation time. It may be accessed sequentially or directly by record number. Relative files can be extended in size subsequent to their creation.

RANDOM FILES

Random files provide a basic organization for those users desiring to manage their own files. Random organization differs from other file organizations as follows:

1. A random file is simply a collection of logically contiguous blocks on a pack set. The number of blocks is specified at the time the file is created and may be expanded dynamically.
2. The user may specify a relative starting block number and a byte count with each read or write.
3. Each write consumes the entire specified block. Bytes not specifically written contain no useful information. The contents of the block include no system information except the block stamp. The one-word block stamp on each granule may or may not be visible to the user (at the user's option). Management of the user's data is the responsibility of that user.

The monitor provides allocation of file space, security checks, and normal I/O queueing service and clean-up. The user is responsible for record management. I-D-S/II uses random files and manages the content.

UNIT-RECORD FILES

Unit-record files are consecutive files that contain unit-record type formatting information such as page heading, vertical format control, and horizontal tab information. These files may be copied to a unit record device (such as a line printer or terminal) and the output will look as if originally written to the device.

FILE ACCESS

When a consecutive file is read, the records are accessed in sequential order. When any other type of file is read, the records can be accessed sequentially or directly. The records are accessed sequentially if no record identifier is specified. The records are accessed directly if an identifier is specified.

A file can be created by writing records sequentially or directly (by specifying an identifier). Existing records can be rewritten in files of any organization. Record positioning operations are allowed in a file of any organization. An error return is taken if the beginning or end of file is encountered, or if a position by record identifier is requested and the record does not exist.

The enqueue/dequeue facility permits simultaneous access to a file by multiple readers or by multiple readers and a single updater. Several user programs executing concurrently in separate jobs may be generating reports from a data file while other user programs are concurrently modifying data items within the file.

Responsibility for coordinating concurrent update activity is divided into two parts, one controlled and provided by the operating system and the other controlled by the application programs via the system's enqueue/dequeue service. The operating system guarantees the physical integrity of the file so that it remains properly connected, regardless of the update activity. It also ensures that readers are provided with the most up-to-date information in response to their requests.

Coordinating logical integrity of the file (primarly the data content) is the responsibility of the application programs, since any connection of the data in one record of a file with that in another record of the same or another file is carried in the application program, not in the file itself.

Applications use the system's enqueue/dequeue facility to gain exclusive access to the records. Enqueue/dequeue is a generalized service and guarantees exclusive or shared access to named items as required and requested. The users of the service must agree on the meaning of the names, e.g., the names of the records containing inventory count.

CP-6 file management includes a type of file sharing which provides a journal facility. Many output users may share a consecutive file (tape or disk) by adding records to the end of the file. No other type of access to the file is permitted as long as the file is open in journal mode. Once closed, the file is no longer a journal and is accessed as any consecutive file is accessed.

RECORD BLOCKING

The system automatically blocks records for other than random files in 1024-word blocks to promote efficient use of disk space. The user does not participate in this blocking and, when reading, will receive the appropriate record within the block, rather than the entire block.

When updating a keyed or indexed file, the user may rewrite a record in a size larger or smaller that the original record size. If necessary, the monitor allocates additional disk space to accommodate a larger size. A record can also be rewritten in a consecutive file but the original record size is maintained. If the new record is larger, the end of the record is lost; if it is smaller, old data remains in the record. Relative file records may be rewritten with any size from zero to the maximum size initially defined for the file.

EFFICIENT FILE TRANSFER (EFT)

The file maintenance processor is called Efficient File Transfer (EFT). EFT provides file protection via disk or tape backup, restore, and archival functions.

BACKUP ON TAPE OR DISK DUALS

User files are copied to some backup medium according to an installation determined schedule. On an individual file basis, the user can specify that a file should or should not be included in the regular backup scheme. Only files thus qualified that also have recently been modified are considered for daily backup. The installation manager can also specify which accounts are available for backup. The backup medium may be either tape or a dual pack set. In general, the installation updates the dual pack set when a set is dismounted. Files may also be backed up on tape while the set is mounted in order to protect active files.

RESTORING FROM BACKUP

The restore function has two applications: to restore an entire pack set because of some major disaster, or to restore individual files which have been damaged or accidentally deleted. If the installation is maintaining pack set duals, the restore-all operation may be as simple as mounting the dual set and creating a new dual. If the backup media include tapes, the process is more complicated since restoration of several sets of tapes may be required to ensure that the latest copies of all files are properly restored.

ARCHIVE

The user may specify that an individual file should be copied to the archive medium ("stowed"). That file is then frozen from updates until it has been stowed. The user may further specify that the file should be put in inactive status (the data is deleted) or active status (the data is retained). Either way, the file remains cataloged and the stow tape identification is retained with the file's information. At some later date, the user may request that the stowed version be returned from the archive medium to the active file system.

PACK SETS

All CP-6 disks are organized into pack sets (i.e., a group of one or more disk packs). All initialized random access devices, whether dismountable or non-dismountable, belong to some pack set. A pack set is identified by its pack set identifier, which is a one to six character name common to all disk packs in the set.

Each member of the pack set also has a unique serial number, used by CP-6 file management when mounting a pack set. External usage of serial numbers is restricted to the system manager or operator for communication with the CP-6 system while initializing, extending, or mounting the set. Users of a pack set are unaware of its serial numbers, referring to the set by pack set name only.

Pack sets in regular use at an installation are normally cataloged by the batch scheduler. Pack sets, cataloged or not, may be mounted as public, private, or exclusive. The only difference is in the mode of mounting; there is no difference in the format. A pack set may be private today and public tomorrow.

Mounting a pack set as public causes the accounts it contains to be merged into the system account directory. The files can be accessed by account: the system automatically recognizes the intended pack set.

Mounting a pack set as private does not enter the accounts on the pack set into the account directory; the files can only be accessed by specifying the pack set name as part of the file identifier. More than one user may reference a private pack set concurrently.

Mounting a pack set as exclusive is similar to private mounting except that only a single user may access the set.

LABELED TAPE

CP-6 supports ANS (American National Standard) labeled tapes. The ANS tape format has two advantages: data protection (unexpired tapes cannot be overwritten) and inter-system portability. Using ANS tapes, data can be transported between the CP-6 system and all other systems which support ANS tapes (other Honeywell systems, IBM systems, etc.).

PROTECTION AND SECURITY

For CP-6 format tape files, information within the header and data records supplies the security features found in the file management system: password and file access control.

Additionally, three mode options allow the system manager to specify how rigidly the ANS protection features are to be applied:

1. Fully-Protected Mode. Only ANS expired tapes which have been initialized by the LABEL processor may be written. No tape serial number specification is allowed at the operator's console and specification of an output serial number forces processing to be done only on a tape already having that serial number.

2. Semi-Protected Mode. A warning is sent to the operator when output is attempted on an unexpired ANS tape. The operator can authorize the overwriting of the tape or overriding of the input and output specification through a key-in.

3. Unprotected Mode. Both unexpired and expired labeled tapes can be overwritten without operator intervention.

TAPE FORMATS

CP-6 supports three groups of labeled tape file formats:

1. ANS Standard formats: F (fixed-length records), D (variable-length records), S (variable-spanned records) and U (undefined). Records in F, D or S format may be blocked on tape; i.e., several data records per physical tape record. Also, the S format file may contain records that span tape blocks. On a U format tape file, each physical record represents a data record.

   All ANS formats are defined to be ASCII data. Format D has 4-byte record headers and Format S has 5-byte record headers that contain the record size in decimal characters.

2. EBCDIC formats: V (variable record length), F (fixed) or U (undefined).

   Records in F or V format may also be blocked. Format V records may also span blocks, and have 4-byte record headers that contain the record size in 8-bit binary.

   Records of EBCDIC format tape files will be translated to ASCII for the user if translation is requested at tape open.

3. All CP-6 file organizations.

   All CP-6 formats are transportable between CP-6 sites. Each tape file contains a File Information Table (FIT) to supply access and control information similar to that contained in a FIT associated with files from pack sets.

The tape file management system can be used to block and unblock, span, and translate tape file records. The user can control volume switching, or it can be done automatically by the system.

INPUT/OUTPUT

All requests for I/O services specify a data control block (DCB) that is used in performing the I/O. The DCB is a data structure used for describing the actual I/O connections and for maintaining information such as the result of an I/O operation. Figure 7-1 illustrates the various connections established for performing I/O. The following points are keyed to the connections illustrated in the figure:

1. The user program references a DCB via a monitor service call.

2. The DCB is connected to one of several types of I/O facilities via monitor services (M$DCB or M$OPEN), or via the IBEX command !SET.

    a. File. A file on a user-available pack set.

    b. ANS Labeled Tape. A file on a labeled tape named as a resource.

    c. Comgroup. A CP-6 logical communications network, commonly used to connect terminals to programs, that connects devices and programs to programs. Through this mechanism, terminals may be accessed by name. (Refer to Section 9, CP-6 Communications Management, for more information.)

    d. Special Name. A name assigned to a logical device. (An example is the destination for listings, e.g., a user terminal (for on-line) or a line printer (for batch)).

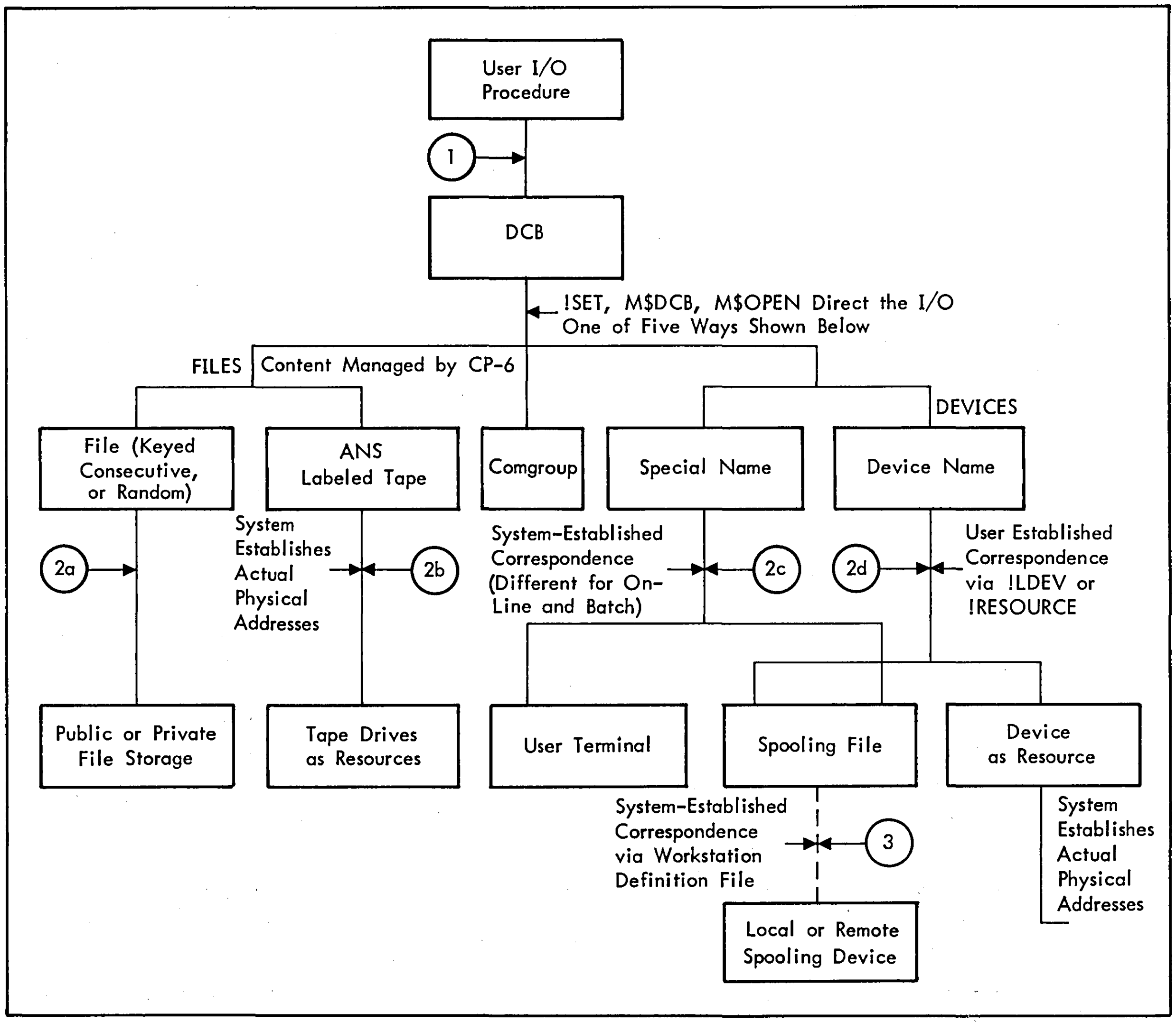
    e. Device Name. A name which is connected to a logical device or a specific device (e.g., a specific tape drive with a foreign tape mounted, which has been defined to be a resource).

3. The contents of a spooling file are copied to a local or remote output device such as a line printer or card punch, or are built by a local or remote input device such as a card reader.

For convenience, a number of available DCBs have default assignments to special names. For example, LO (listing output) is the assignment of the M$LO DCB, and CR (command stream read) is the assignment of the M$SI DCB.

These DCBs are set up for common system usage.

Each request for I/O service from the monitor is made by inclusion of an I/O call in the user's program. This call references a Function Parameter Table (FPT), which in turn refers to a Data Control Block (DCB). The combination of the I/O call, the FPT, and the DCB provides the information that the monitor needs to perform the requested operation.



User I/O Procedure
1
DCB
!SET, MDCB, MOPEN Direct the I/O One of Five Ways Shown Below
FILES
Content Managed by CP-6
DEVICES
File (Keyed Consecutive, or Random)
ANS Labeled Tape
Comgroup
Special Name
Device Name
2a
System Establishes Actual Physical Addresses
2b
System-Established Correspondence (Different for On-Line and Batch)
2c
2d
User Established Correspondence via !LDEV or !RESOURCE
Public or Private File Storage
Tape Drives as Resources
User Terminal
Spooling File
Device as Resource
System-Established Correspondence via Workstation Definition File
3
System Establishes Actual Physical Addresses
Local or Remote Spooling Device

Figure 7-1. Connection Established for Performing I/O

DEVICE INPUT/OUTPUT

The CP-6 system supports four classes of I/O devices in addition to files and labeled tape:

1. Interactive terminals.

2. Comgroups.

3. Unit record peripherals (local or remote).

4. Unformatted devices.

5. Formatted devices.

A DCB used for I/O by a user program can become connected to these types of devices by assigning a DCB to a special name or device name.

Special names are a set of convenient default assignments which are set up on a system basis with one set of assignments for on-line jobs and one set for batch jobs. These assignments are to logical devices or to the interactive terminal.

A device name may be created and its output directed to unit record peripherals by the LDEV command, or a name may be created and associated with a formatted or unformatted device by the RESOURCE command. A third type of connection to unit record peripherals may be made by specifying a device name and work station directly in the DCB.

INTERACTIVE TERMINALS

Use of terminals as I/O devices is highly flexible. Terminals can be operated in echoplex mode or with local printing. Input features include sophisticated editing and tabbing capabilities. When operating in echoplex mode, typeahead is allowed and proper sequencing of input and output is guaranteed. Output features include tabbing, line breakup to fit the device, pagination, and page headers. A wide variety of timing algorithms and code sets are provided to fit the idiosyncracies of specific terminals.

UNIT RECORD PERIPHERALS

All I/O to unit record peripherals (e.g., card readers, card punches, line printers, and plotters) is staged via spooling files whether the device is local or remote. Spooling means that the entire input job from a card reader is read to disk before processing, and that all of the job's output to these devices is stored on disk and is normally not output until the job is complete. (Spooling is discussed in more detail in Section 13.)

The benefits that accrue from processing of these devices symbiotically include:

- Program execution is disconnected from I/O devices.
- Smoothed peaks and valleys in I/O demand.
- Multiple programs can be output to the same device simultaneously.
- Output can be grouped by form type.
- A program can generate several 'streams' of output to one device.
- Several copies of output can be produced.
- Batch peripherals can be used on-line.
- Jobs can be submitted to the batch job queue from on-line terminals.
- Job requirements can be pre-scanned for efficient resource allocation in batch scheduling.

UNFORMATTED DEVICES

An unformatted device (primarily foreign tape) is handled as a resource which must be pre-allocated (contended for if on-line). When the device is allocated to the user, he or she is responsible for the data read or written to the device. No blocking or formatting services are provided.

FORMATTED DEVICES

If one of the labeled tape formats is specified when using a tape device, record formatting will be done for the user by tape file management. A formatted tape may contain blocked and spanned records, and may have data translated from EBCDIC to ASCII and vice versa.

LOGICAL DEVICES

A logical device is an information stream through which a user may conveniently perform I/O. A logical device may be associated with any physical device that the user specifies, provided that the device is a spooling device. (Spooling devices include unit record devices such as the line printer, card reader, card punch, and all devices that are accessed via remote processing.)

Logical devices may be defined by the user by means of the LDEV command. Two logical devices are available for use without specific user definition; they are used for line printer output and card punch output.

Logical devices perform three major services. First, they may be used to merge (or separate) output from different user DCBs intended for the same destination device. Second, their names may serve as a convenient shorthand for an entire set of device characteristics (for example, device name, form name for printer, number of copies, etc.). Third, the two predefined streams allow users to run jobs with little concern for the physical location or connection of devices on the system.

One important use of logical devices is production of several separate reports on a single program pass. Output for each distinct report is directed to its own logical device. Each logical device is separately buffered on disk file and, on completion, the reports are transferred to printer either serially (if there is a single printer), or in parallel (if more than one appropriate printer is available).

FEATURES OF THE FILE SYSTEM

In addition to the facilities of the file system outlined above, CP-6 file management provides the following special features:

- Unique block stamps on disk storage blocks provide not only security but maintainability by providing information to aid in the reconstruction of file(s) in the event of a major disaster.

- Directory blocks are linked and the forward and backward links are checked on access.

- At job step and during recovery, the user's open files are closed properly and current updates are posted to the files.

- Pack sets provide a separation of files in such a manner that a disaster is localized and may affect a file or an account but will not affect the entire file system. Full system restores are not necessary and thus availability of the system is high.

- A check-write feature for disk I/O is provided as an installation option for file directories or all granules.

- A user may change the account used as a default by file management without logging on or off.

- File space is allocated by extents (groups of blocks) rather than by single block. This is beneficial in two ways. First, damage to several blocks may be confined to damage of one file rather than several. Second, a reconstruction of the directory or cataloging information is fast, since only the FIT of each file needs to be read in order to reconstruct the entire set.

- Each file extension is updated immediately in the FIT so that memory failure cannot cause the loss of file blocks.

- Entire files and individual records can be encrypted and decrypted.

- File directory descriptors and dates are updated when the file is opened so that no recovery action is required.

- If a file was opened and for some reason (e.g., a major disaster) was not closed, the recovery process will reconstruct the necessary control information.

- An option is available to determine the next account in the account directory and the next file in a file directory provided the user has proper authorization.

- Data compression and automatic decompression may be requested by the user as a file attribute.

- Write-ahead and I/O cache are implementation details that enhance performance; the user need not worry about buffering efficiency.

- Data intended for a printing device may be sent to a file for the user's inspection and later sent to the printing device with the device information intact (e.g., forms and vertical format control).

- File attributes include a code which is used by programs to determine the type: APL workspace, various FORTRAN output formats, BASIC programs, etc.

- In most cases, files are automatically extended in size as the file grows. However, random and relative files are extended by specific request rather than by automatic extension.

Section 8

CP-6 Scheduling and Memory Management

The CP-6 scheduler provides rapid throughput and on-line response. CP-6 memory management provides a highly flexible and secure computing environment. Shared processors and libraries save memory space by associating users with common routines. The system automatically detects and shares concurrently used programs. An installation can, by instituting its own shared entities, further reduce over-all memory costs.

SCHEDULING

A vital part of the CP-6 operating system is the scheduler, a module responsible for allocating the central processor (as a resource) to various jobs. The scheduler provides fast terminal response to on-line users and rapid throughput for all jobs. The degree of efficiency with which the scheduler performs its role is the key to optimum utilization of a computer system -- and the value of the computer to an organization.

The CP-6 scheduler performs two major functions in achieving this goal of high performance:

- Selecting the highest priority job that is ready for execution.
- Ensuring that the remaining high priority jobs are ready to use the processing resource when it becomes available.

The CP-6 scheduler accomplishes this by:

- Creating prioritized status queues in which every job has a single entry.
- Assigning a priority to every job in the system.
- Modifying a job's priority as requirements for access to the processing resource change in response to events triggered during the programs execution (such as I/O and terminal input).

There are three fundamental classes into which the various status queues may be segmented:

- Waiting to Execute. This group of queues contains those jobs requiring a processing resource in order to proceed.
- Executing. This queue consists of a single entry for each central processor: the job currently using the processing resource.
- Non-Executable. This group of queues contains jobs waiting for an 'event' to occur before requiring access to the processing resources (e.g., completion of an I/O operation or availability of a system resource).

A primary benefit of the priority queue structure is that it provides complete service to I/O users while concurrently keeping the processing resource busy with compute-bound jobs. This feature enables maximum utilization of both I/O devices and the central processor.

Each job is assigned a base priority when first activated. The base priority may be different depending upon the selected mode of operation - for example, batch or on-line. This feature allows one class of jobs to gain preferential service. Under normal operation, the priority of a job changes frequently during processing. Conditions or events that cause the scheduler to modify a job's priority include:

- Completion of an I/O operation.
- Completion of terminal input.
- Occurrence of an error during an execution program.

The executing programs and the environment continually notify the scheduler of their requirements and of the availability of resources. As a result, the scheduler can efficiently and effectively optimize the entire system. Dynamic system tuning is a major factor in making the CP-6 system an efficient multi-use operating system.

Another mechanism used by the CP-6 scheduler to increase the amount of time spent in processing user jobs is the use of bounded time intervals. The system-control parameters QUAN and QMIN are time intervals. They are set to ensure that no user job receives more than its share of the processing and memory resources, yet still gets enough to continue processing efficiently.

- QUAN is the maximum time-slice allowed a compute-bound user before another job is given control of the system. This assures that no single compute-bound job of a given priority can dominate the processor resource. The QUAN value is separately specified for each batch stream and all on-line users.
- QMIN is the amount of processor time guaranteed a job unless pre-empted by a critical task. The processor will still respond to I/O interrupts and perform other system tasks. But, the processor will not be given to another user until the current user has received its QMIN quantum.

VIRTUAL MEMORY AND SECURITY

The Virtual Memory and Security feature provides virtual memory addressing and protection capabilities at the hardware level. The virtual memory feature is based on the concept of working spaces and segments within working spaces rather than upon a 'demand page' basis. The security aspect is grounded in this virtual memory management concept. The paragraphs that follow describe the CP-6 system implementation of virtual memory and security.

The virtual memory concept allows a large virtual address space to be divided into Working Spaces (WS). In the CP-6 system, these workspaces are limited to one million words each. A WS is further divided into variable size independent spaces called segments. A segment is defined by at least one descriptor which locates that segment in virtual memory. The descriptor indicates the WS in which a segment resides, the base address of the segment relative to the WS, the size of the segment, and the access allowed to that segment.

To reference any portion of virtual memory, a procedure must have a segment descriptor which frames the particular area and which gives the desired permission (e.g., read, write and/or execute).

Effective addresses (the result of normal address arithmetic) are segment relative. An effective address is converted to a WS relative address by adding the base address of the segment, as defined by the segment descriptor, to the effective address.

Each WS is divided into 1024-word pages. Each WS has a Page Table that identifies the physical pages allocated to the WS and the access allowed to each page. The associated page table is located by using the working space number as an index into the Working Space Page Table Directory (WSPTD). The WSPTD is simply a table whose entries are pointers to each WS page table. The WSPTD itself is located via the Page Directory Base Register (PDBR). This mapping process is illustrated in Figure 8-1.

A domain is defined by the segments it may access and the access right of those segments. The segments need not be contiguous and may encompass more than one working space. A domain can reference only those areas of virtual memory framed by the segment descriptors which are available to the domain. The user cannot create a segment descriptor, or change the location or increase the size of the area originally framed by the monitor-prepared segment descriptors. Figure 8-2 shows two simple domains on a user's working space: that of a user program and that of the operating system.

There are two types of segments: operand segments and descriptor segments. Operand segments contain instructions, data, or a combination of both. Descriptor segments contain only descriptors.

The Virtual Memory and Security feature provides several levels of isolation and protection. At the first level, everything is accessed via a descriptor which directly or indirectly (via a Work Space Register) addresses a WS and provides a limited window into that WS. At the second level, the WSPTD specifies whether or not the WS exists (by a flag which signifies presence or absence of a page table). The third level is provided by the domain concept. To reference a segment, a process must have a descriptor for the segment. The reference must be within the virtual area framed by that descriptor and it must be consistent with the permission granted by that descriptor. Figure 8-2 illustrates the relationships between descriptors and the WS.

Another major factor contributing to the protection mechanism is that a slave mode user is prohibited from modifying any Work Space Register, the Linkage Segment Register, and the contents of the linkage segment. The slave user is also prohibited from addressing in the absolute mode or manipulating the page tables.

CP-6 supports the following five levels of execution for each user, each with an established priority and its own domains of reference (areas of memory to which a process has access):

1. User (USR).

2. Alternate Shared Library (ASL).

3. Interactive Debugger (IDB).

4. Command Processor (CP).

5. Monitor (MON).

Figure 8-3 illustrates these five domains together with the paths of control transfer which exist between them.

Each level of execution has its own well-defined domain, which is granted by the monitor. While many users will be resident at the same time, scattered over the real memory available, the Virtual Memory and Security feature prevents any user from accessing the memory of other users or special shared processors.

The CLIMB instruction transfers control to another domain and saves the environment of the calling domain on the Safe-Store Stack (SSS). The called domain can then return to the domain from which it was called by execution of the return form of the CLIMB instruction (which restores the environment from the SSS).

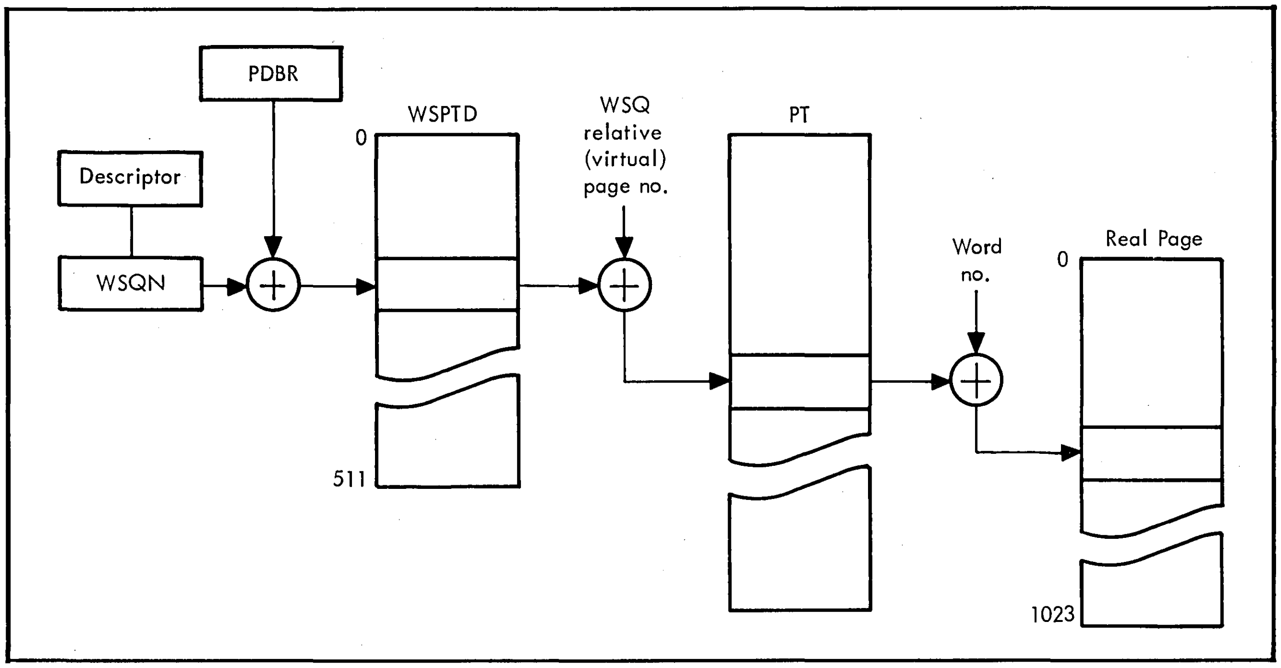


Figure 8-1. Memory Mapping

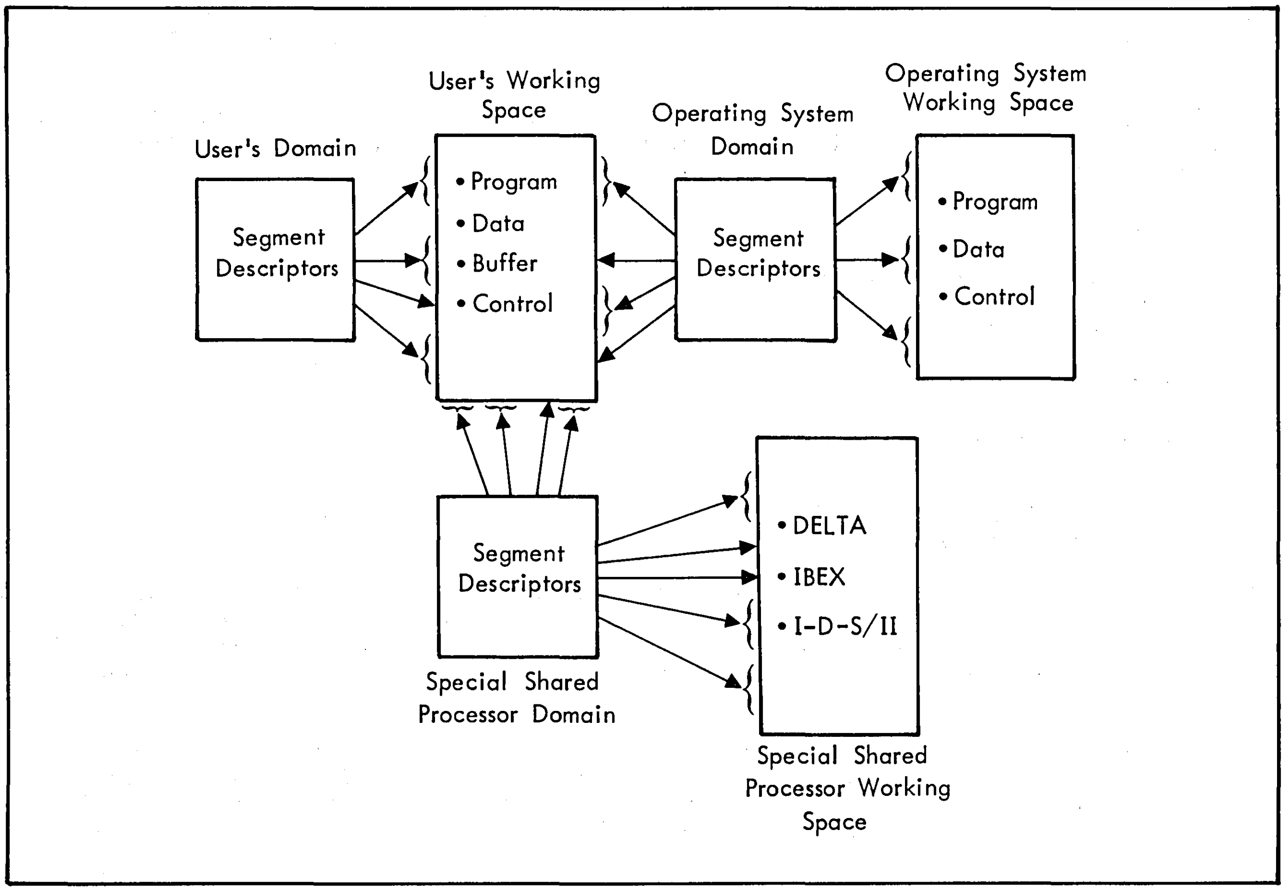


Figure 8-2. CP-6 Domains of Reference

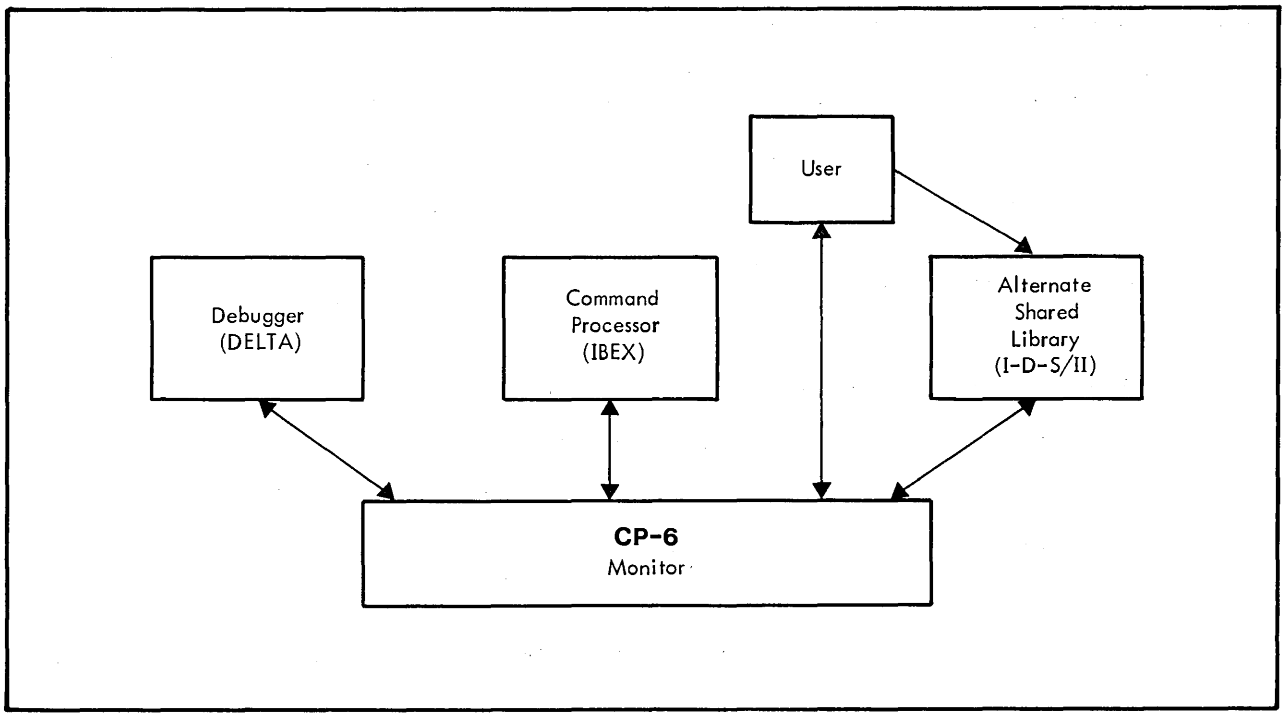


Figure 8-3. Control Paths Between CP-6 Working Space

The basic approach to CP-6 memory allocation and execution control includes the following technical features:

- The CP-6 system uses only one 64-word Safe-Store Stack frame per domain.
- A program may enter a domain at a higher level of privilege or priority only via a CLIMB instruction using entry descriptors controlled by the monitor.
- Exit from a higher level domain restores the privilege and priority to those that existed prior to entry.
- The monitor resides in its own working space.
- The procedure of the monitor is protected by placing it on housekeeping pages.
- Each user has a unique working space.
- Up to three special shared processors may be associated with any given user: an alternate shared library, a debugger, and the command processor. Each resides in a separate working space.
- Ordinary processors with shared procedures execute within the user's working space. These include most language processors such as APL, FORTRAN, BASIC, and RPG II.
- Working spaces are organized in the following manner:
  - Real addressing can only be referenced by monitor procedures running in privileged master mode.
  - Interactive command processors, debuggers, and alternate shared libraries are procedure only.
  - Each user's working space contains all pages that are assigned to him.

USER VIRTUAL MEMORY LAYOUT

The user's megaword of virtual space includes the following segments:

1. Job Information Table (JIT)

   The JIT contains accounting information for the user's working space.

2. The automatic storage stack (TSTACK) used by the monitor when processing on behalf of the user.

3. Housekeeping JIT (HJIT)

   The HJIT includes the following segments:

   - The Linkage Segments (LS), defining user domains of reference.
   - The Safe-Store Stack (SSS), for saving interrupt environments.
   - The Parameter Stack (PS), for passing parameters between routines.

4. File Buffers.

   All file buffers are allocated from a common pool within this segment.

5. Special Shared Processor Data Segments.

   Space for data required by a debugger, alternate shared library, or interactive command processor is allocated from this area.

6. Data Control Blocks (DCB)

   The DCB contains information used by the monitor to perform I/O operations for the user.

7. The User's Instruction Segment (IS)

   This area provides a 256K-word area for the user program (instructions and compiled data) and dynamic data. If the program requires the use of a run-time library, the user program is restricted to 224K words.

8. User Data Segment

   This area can contain up to eight independent user data segments, the first two of which are used for PL-6 automatic data and COMMON data. A total of 384 words are available in this area.

Figure 8-4 illustrates the layout of the user's virtual space within his working space. With the exception of a fixed minimum requirements for HJIT, JIT, DCBs and buffers, user physical pages are demand allocated.

SHARED PROCESSOR FACILITIES

The CP-6 shared processor facilities share automatically all user-domain procedure from non-overlaid run units. Libraries and overlaid programs and processors which do not execute in the user's domain (e.g., command language processors, debuggers, and data base managers), may also be shared by explicitly declaring them to the system, either at initialization time or while the system is in operation. Each user of a shared processor has a private copy of only the data and DCB portion of that program; the procedure (code) portion is shared by all users associated with the program.

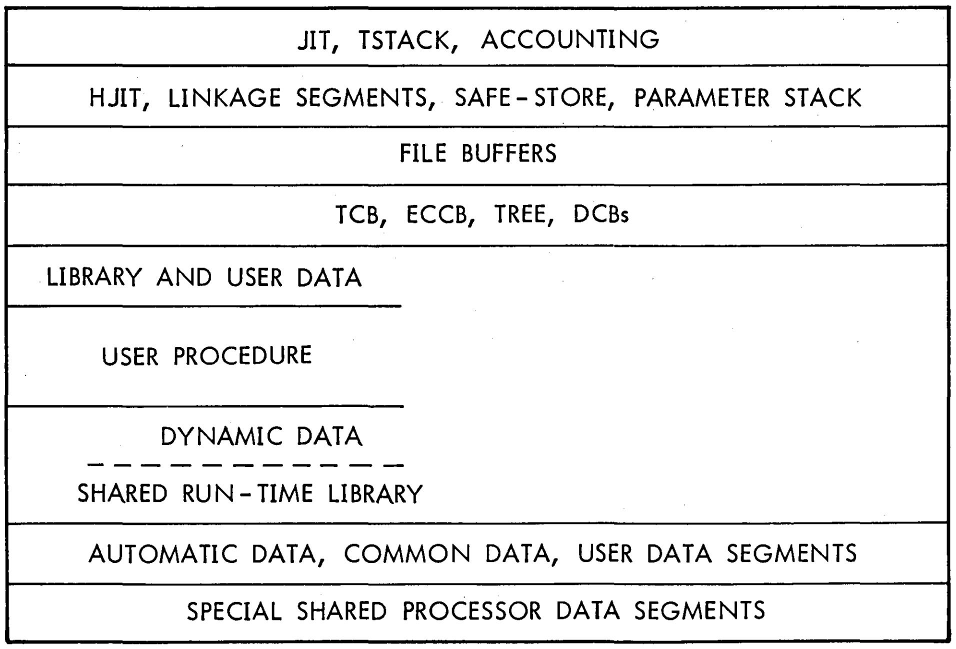

JIT, TSTACK, ACCOUNTING
HJIT, LINKAGE SEGMENTS, SAFE-STORE, PARAMETER STACK
FILE BUFFERS
TCB, ECCB, TREE, DCBs
LIBRARY AND USER DATA
USER PROCEDURE
DYNAMIC DATA
SHARED RUN-TIME LIBRARY
AUTOMATIC DATA, COMMON DATA, USER DATA SEGMENTS
SPECIAL SHARED PROCESSOR DATA SEGMENTS

Figure 8-4. User's Virtual Memory - Megaword Working Spaces (not to scale)

The automatic sharing feature will make the best possible use of memory, dynamically tailoring the system to maintain only one copy of a particular program in memory, without prior knowledge of that program's probability of common usage.

CP-6 supports the following three types of shared processors:

- Standard shared processors.
- Run-time libraries.
- Special shared processors.

STANDARD SHARED PROCESSORS

The term "standard shared processor" is nearly synonomous with "automatically shared program". A standard shared processor is any user program created by the LINK processor that has at most one level of overlaying and contains only pure procedure. A standard shared processor resides in the user's working space, and may have initial data and DCBs which will be unique for every user. The procedure portion of the processor will be shared by all associated users by mapping that procedure into each user's working space.

A non-overlaid program will be automatically shared when it is concurrently accessed by more than one user, unless either the LINK processor has built an unsharable program or a user starts the program under a debugger.

Overlaid standard shared processors must be explicitly declared, either at system initialization time or while the system is running.

SHARED RUN-TIME LIBRARIES

A shared run-time library is a collection of frequently used subroutines which is treated by the CP-6 system in such a way that multiple programs may simultaneously use the same copy of the library, resulting in efficient use of main memory. A number of public libraries are supplied with the CP-6 system (e.g., the FORTRAN Run-Time Package and the COBOL library). User installations may create additional public libraries which suit their specific requirements.

Run-time libraries are shared by all simultaneous users associated with the library by having the library procedures mapped into the top 32K of each user's instruction segment. Data required by the libraries is supplied individually to each user.

A call to a run-time library does not cause a change of domains.

SPECIAL SHARED PROCESSORS

The CP-6 system recognizes three types of special shared processors:

- Alternate shared libraries.
- Debuggers.
- Command processors.

The standard CP-6 system includes I-D-S/II as the supplied alternate shared library, DELTA as the debugger for FORTRAN, COBOL, PL-6, and assembly language, and IBEX as the interactive and batch command processor. Installations commonly add to the standard CP-6 special shared processor other special shared processors for their own use in any of the three catagories.

The procedure portion of a special shared processor resides in it own working space. The procedure portion is shared by all users associated with the processor by referencing the processor's unique working space. A data area is provided in each user's own working space for each special shared processor.

A call to a special shared processor causes a change of domain, thereby changing the areas of memory to which the processor has access. The areas of memory to which the processor has access are determined by its type.

Special shared processors must employ dynamic data segments for all their non-constant data. As with run-time libraries, special shared processors must be self-contained.

ALTERNATE SHARED LIBRARIES

The alternate shared library provides I-D-S/II with an environment that allows file access protection, data protection, and greater control of buffers during error recovery operations.

The procedure portion of an alternate shared library resides in its own working space. By locating I-D-S/II outside of the user working space, it is possible to identify its calls and thus allow file protection by excluding access except by I-D-S/II. The context and buffers reside in the user's working space because they are unique to the user.

A user program that calls an alternate shared library relinquishes control until the library returns control to the user. User-established break control, timer runout, and event reporting are deferred while the library is in control.

DEBUGGERS

The system-supplied debugger is known as DELTA. DELTA can access everything that the user can, but is not allowed to access procedure and data for other special shared processors. In addition, DELTA has its own working space and thereby does not occupy any of the user's virtual memory. Descriptors in DELTA's linkage segment provide full access to all segments within the user's working space to which the user has access, to DELTA's procedure in its own working space, and to the debugger data area.

COMMAND PROCESSORS

The CP-6 supplied command processor is called IBEX (Interactive and Batch Executive). It consists of pure procedure that resides in the working space reserved for command processors. It requires only limited access to user's working space, JIT, DCBs, and command processor data segments. In addition to processing execution control commands, IBEX recognizes and processes calls to shared processors (e.g., FORTRAN, BASIC, APL, COBOL) and fields 'interrupts' from time-sharing terminals, allowing the operator to quit, continue, or invoke DELTA at the point of interrupt.

User-written command processors reside in the command processor working space concurrently with IBEX and with one another. Only one command processor can be associated with a given user at a given point in time.

CP-6 Communications Management

CP-6 communications management provides cost-effective communications via local and remote communications processors, each of which contains the software required to interface all supported types of terminals to the network. CP-6 communication management allows the system manager to configure and attach devices so that communications are tailored to installation requirements. Devices are accessed as time-sharing and peripheral devices and via comgroups. Comgroups provide a vital communication link between programs, and between programs and devices. A unified set of virtual protocols support access to time-sharing, CRT and graphic terminals; to unit records devices; via forms and between user programs.

COMMUNICATION PROCESSING

Communications configurations may be geographically distributed and connected to several CP-6 host processors as well as to a variety of terminals. Figure 9-1 shows a sample CP-6 communication configuration.

Communications capabilities on the CP-6 system are provided on Honeywell mini-computers. There are two types of communications processors: local (CP) and remote (RCP).

CPs are directly connected to a host processor via a direct I/O connection which can transfer up to one million bytes per second between CP memory and host memory. RCPs are connected via commercial carrier lines to CPs or to other RCPs. These lines may be dial-up or private and may be any speed required by the load (up to a maximum of 72K bits per second). Higher capacity connections may be obtained by using multiple parallel paths between CPs and RCPs. Multiple paths can also be used to increase reliability of connection. The CP-6 system uses a full-duplex bit-oriented protocol over these lines for maximum throughput and control over errors. The line protocol is Honeywell's HDLC which is similar to IBM's SDLC; X.25 protocols form the next higher level of communication, enabling connection to (or via) many standard networks around the world.

Each communications processor, whether local or remote, contains the software required to interface all supported types of terminals to the network. The supported types of terminals include asynchronous terminals (such as CRTs), synchronous terminals on either clustered or multi-dropped lines, remote batch terminals, and HASP-protocol terminals. The interface software transforms the characteristics of each of the supported terminals into the CP-6 standard protocol, permitting freedom in programming from the characteristics of individual devices.

The communications subsystems are logically independent of the hosts that connect to them, but are physically dependent on one or more hosts for initialization, log-on files, and many other services. Once initialized, however, the communications capability continues to be provided during times when one or all of its hosts are out of service, allowing the user to recover from where he left off (within certain limits) when the host returns.

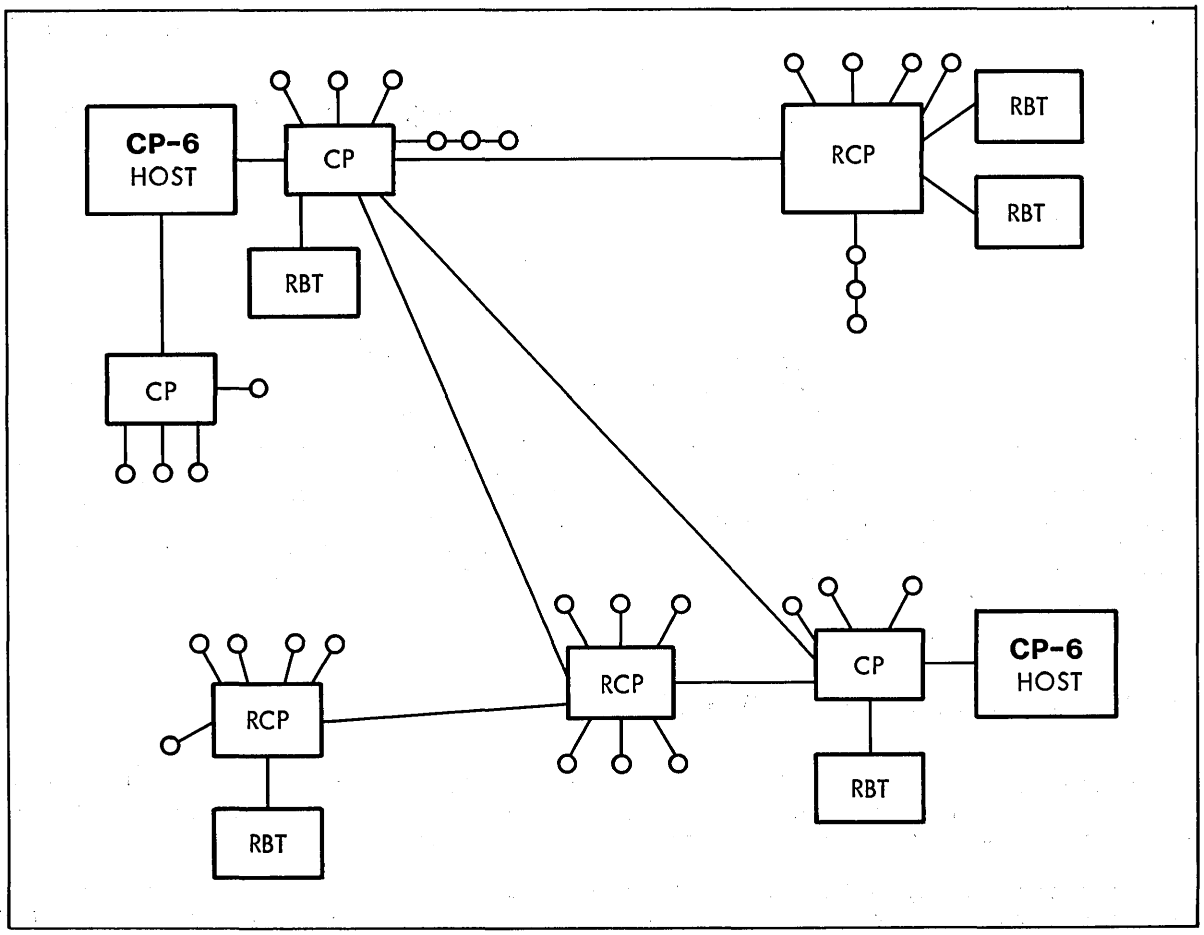


Figure 9-1. Sample CP-6 Communication Configuration

CP-6 hosts can connect together for purposes of remote job entry, job load leveling, file transfer, and data record access.

CONNECTING TERMINALS TO PROGRAMS

In the CP-6 system, the terminal or device is the unit of allocation; that is, each terminal is separately connected to a program even if the terminal is one of many on a multi-drop line. CP-6 communications processing, not the user program, controls polling on such lines. This feature isolates applications from device-specific characteristics and makes it possible, for example, to use one of several multi-drop terminals on a single line for time-sharing while the others are connected to a transaction processing application.

Similarly, each device associated with a remote processing terminal is separately allocated. A CP-6 station is not a description of a single physical device, but rather a logical collection of physical devices. This kind of station extends to the local peripherals of the host so that a logical workstation may be made up of a particular printer and a particular card reader on the local I/O multiplexor (IOM). Such a configuration is often useful for student programmers, for example, whose output must be delivered to the printer adjacent to the card reader used for the job.

Terminals log on to the CP-6 system in two stages:

1. Communications log-on.

2. Host log-on.

However, the two stages are usually not visible to the user.

Communications log-on is controlled by the contents of the SUPER file which is maintained by the Communications Manager. When a terminal connects to the configuration, the log-on identification is collected and supplied to the communications log-on process over a previously defined path. Information returned from communication log-on defines the characteristics of the caller using previously stored information, such as the exact device complement of a particular remote batch terminal (RBT), and to which host connection should be made. Connections are set up between each terminal device and the appropriate host. The system can supply default log-ons by line number to permit log-on of plotters and other devices which do not have transmission capability.

This log-on process dynamically establishes the proper parameters for the terminal. Log-on information is created dynamically during system operation and thus may be changed without any sort of system definition (SYSGEN) process. The CP-6 communication configuration and capabilities are adjusted and augmented during system operation rather than by an off-line reconfiguration process. Terminals are connected to a particular host during the log-on process, either automatically or at the user's explicit request.

Note that all connections are established dynamically at the time a terminal calls and identifies itself and that these connections are under the control of the Communication Manager via modifications to the contents of his SUPER file. No 'SYSGEN' or initialization process is required except that which establishes the path to communications log-on. Shifting activity from one host to another is simply a matter of modifications to the SUPER file.

The connection process also connects programs in one host to programs in another host. Both programs may be within CP-6 hosts, both may be within CP-6 real-time hosts, or they may be combined. Thus the needs of real-time programs for communication with each other are served by the same facilities that service terminal communication.

CONFIGURING, ATTACHING AND ACCESSING COMMUNICATIONS DEVICES

Control over the devices connected to a CP-6 system is managed at three levels.

1. Configuration of devices into groups called stations.

2. Attachment of the devices to the system via log-on.

3. Accessing those devices from user programs in one of several convenient ways provided by CP-6.

Each of these levels is managed dynamically. Devices and terminals may be configured, attached, and accessed while the system is in operation. Furthermore, the configuration, attachment and access is location and network independent, allowing devices to be used in appropriate logical ways regardless of physical location.

Configuration defines a set of devices which make geographic sense (such as the printers and card equipment in a particular room or building). This 'station' is controlled by an operator terminal designated in the station configuration.

Configuration is logical; that is, the grouping of devices is not constrained by the physical association of devices (for example, the printers and card readers on an RBT). The system disassociates each device on an RBT and then permits the system manager to create logical stations from devices according to need and usage rather than physical connection. Thus two output-only serial printers, a plotter and a terminal, may be grouped into a station and addressed as such.

Attachment defines how devices are connected to the system. There are three ways to attach devices:

1. As a time-sharing terminal. This connection enables the control stream of time-sharing jobs and the workstation-of-origin for time-sharing users. It associates the user with the set of peripherals -- and thus, usually, the physical location. The workstation-of-origin is associated with a time-sharing user by log-on default or explicit command. Typically, a set of time-sharing users will have nearby printers to which their printed output is directed automatically by the associated workstation-of-origin.

2. As a private resource attached directly and exclusively to a single program. Tape units are the principal example of this kind of connection.

3. As a member of a comgroup. Terminals associated with a particular transaction processing application set are connected this way. Comgroup attachment provides wide flexibility both in selecting the number of terminals connected to a transaction processing program, and in the number of processes serving a particular transaction type. Spooled symbiont printers and unit record devices are also connected via comgroups.

The CP-6 system provides three ways of dealing with devices or groups of devices:

1. As time-sharing terminals.

2. Via workstations.

3. As comgroups.

The way the device is accessed is affected by the method of addressing the device.

The time-shared terminal is addressed by default. The programmer need never explicitly direct I/O to his or her terminal. The system puts reasonable output on the terminal and receives input from the terminal unless directed otherwise.

Workstation addressing provides an abstract addressing to a type of device at a particular station, usually at a geographic place. Thus, the user may address "a line printer at Boston" or "a plotter in Bermuda". This kind of addressing is strongly associated with remote batch terminals. But, the CP-6 system has more flexibility of physical and logical configuration; access may come from time-shared as well as batch programs.

Comgroups enable networks of devices (often terminals) in which the addressing is direct. Programs communicate with terminals on the comgroup network by naming the terminal desired just as one names the terminal desired on a multi-drop communication line. Again, the CP-6 system has separated the physical connection from the logical connection, permitting, for example, two terminals connected to a single physical multi-drop line to appear on separate comgroups, if desired.

Thus, access to the devices may be made in three ways:

1. As time-sharing devices. Time-sharing access provides the control access path to time-sharing programs. These programs are logically associated with the workstation-of-origin, but I/O may be explicitly directed to any named workstation.

2. As peripheral devices, typically printers and card readers. Access may be direct or spooled. All peripheral devices are grouped into logically associated collections called workstations. I/O is directed by workstation name, and no physical association of devices is implied or required even though it may be typical.

3. As terminals or devices accessed via comgroups -- the private logical association of programs, terminals, and devices.

COMMUNICATION PROTOCOLS

The CP-6 system includes a set of virtual protocols or access methods. Each access method may be thought of as the description of the features of a 'virtual terminal' and how to use them. The access methods are unified so that a user can either write programs which work correctly regardless of the access method and end device, or can use an access method that takes full advantage of the capabilities of a particular terminal class. The available CP-6 access methods are:

1. Terminal. The device looks like a time-sharing terminal. Operations are available to set prompting strings, change the platen width, etc. Minimal formatting is done. Vertical format control and tabs apply. A TRANSPARENT option is provided to send and receive byte strings to the device without system intervention.

2. Unit Record. The device looks like a line printer, card punch, or card reader. Printer page formatting and card sequencing apply. All output data is translated into printable graphics unless this feature is explicitly suppressed.

3. Forms. The device appears to accept the name of a form and formats all data based on it. Input and output are simply character strings interpreted and presented on the device by the named form. The user's program has no presentation responsibility.

4. CRT. The device appears to be a modern CRT terminal with such features as cursor control, highlight and blink.

5. Graphics. The device appears to be a modern graphics terminal.

6. Inter-user. Features unformatted conversation between user programs with facilities to, for example, get the attention of the other programs.

Note that in choosing an access method the user does not change the operations used to converse: READ, WRITE, OPEN, and CLOSE. Exactly what these operations do varies by access method, just as they do by file type (e.g., keyed and consecutive files). But, programming can be done in such a way that such variations are of no concern. Additional operations are available in some access methods to control special terminal features. These, however, are 'device independent', in that the system always extracts the meaning appropriate for a particular device even if the meaning is simply ignoring the command.

All access methods are convertible to the needs of each type of terminal, because every access method is interfaced to the common protocol. At the destination terminal, this protocol is transformed for the needs of the specific terminal, ensuring a standard method to provide interfaces for new terminals as they are developed. This feature also permits connection to the generalized protocols of other networks.

COMMUNICATION GROUPS

Certain teleprocessing applications require a gathering of terminals into identifiable groups. These applications are particularly common in transaction processing. For this purpose, the CP-6 system provides private networks of terminals called Communication Groups (comgroups) which have the following properties:

- A DCB can connect a program to many devices or terminals in a comgroup.
- Many separate programs (CP-6 jobs or processes) can connect to a comgroup and each may have an address on the group. This provides 'multiprogramming depth' for processing of a single transaction type by shared procedure programs.
- A special read operation delivers the next message (arriving from any terminal in the group) to the reading program.
- The normal write operation delivers a response to the terminal which supplied the input without requiring the transaction program to be aware of terminal addresses or names.
- The group can be composed of devices or terminals from anywhere in the CP-6 network, unrestricted by differing physical characteristics.
- Terminals can be dynamically joined to and removed from the group.
- An optional file-backed queue of messages is associated with each comgroup in which messages awaiting processing may be stored.
- The group is controlled by an administrative user who permits access to the group, directs transaction handling, sets priorities, and controls multi-programming depth.

Communication groups are used by the CP-6 system for input and output symbiont terminals and operator console terminals.

RECOVERY

When a host suffers a temporary system halt, the communications subsystem (CPs and RCPs) sustain themselves and ride through the period, minimizing the effect on connected users. The effect of such an interrupt as seen by the user is dependent on the terminal type and the system options selected. For example:

- Terminals connected to one host will not be affected if another host crashes.
- A crash of a single machine in the system is automatically recovered. Users of other parts of the system are unaffected.
- Lines between network nodes may be added, deleted, or recalled without loss of data when re-establishing reliable communications.

Diagnostics and dynamic verification programs can isolate faults in parts of the system to the extent that the machine is 'well enough' to execute diagnostics. Verification procedures may be run periodically to check and report on the status of communication lines, thus making faulty lines visible to customer engineers. Errors are reported to error logs where they form a profile useful in predicting potential trouble areas or lines.

Terminal devices may be added to the system during system operation. No communications shutdown is necessary to add the capability for an additional terminal. Because the software is capable of adding programs dynamically by down-line loading them from a host, the system may add handlers to accommodate a new terminal type without interruption.

Section 10

CP-6 Reliability and Security

The CP-6 system is designed to be thoroughly reliable and secure. On-line diagnostics, error-tracking, and an efficient recovery procedure result in a system with a minimum of down-time. CP-6 security features promote an environment suitable for the handling of several levels of classified information.

RELIABILITY

Errors are logged into buffers in main memory, which are copied to the system log file. In addition to error condition records, the system log file contains a number of information-only records that include information on tape mounts and dismounts, operator input, and firmware loads.

The system log file is listed and summarized through the system log listing processor (ELAN). ELAN lists and sorts the system log file. ELAN output furnishes a meaningful, comprehensive diagnostic evaluation of the system and its peripherals, aiding in the early detection of potential component failures and thus increasing the reliability, maintainability, and availability of the system.

The error file is also available for on-line preventive maintenance of the system and for diagnosis and prediction of hardware malfunctions.

ERROR THRESHOLD REPORTS

The system accumulates hardware error rates over time (including those successfully recovered) and issues reports to the field engineer when these rates exceed a prespecified value. These reports direct the attention of the field engineer to those portions of hardware which are failing at abnormally high rates.

ON-LINE PERIPHERAL DIAGNOSTICS

Within the system, diagnostics are provided that may be used from either local or remote terminals to analyze the performance of card readers, card punches, line printers, magnetic tape drives, and disk drives. These diagnostic programs run during system operation without disturbing on-line users or batch job throughput (except, of course, for jobs requiring the downed device). Full direct access to the device is provided, and all hardware status information for the specified operation is returned to the diagnostic program.

RECOVERY

CP-6 recovery features attempt to make the system available as much as possible with minimal loss of data when problems occur. A recovery package is offered that takes actions based on the seriousness of any problem that occurs. The resulting recovery is completely automatic, requiring operator intervention only for the most serious problems (such as power interruption).

The various modules of the CP-6 system check the consistency of the resident operating system tables and the important user context. If an inconsistency is detected, or if a hardware error is reported which compromises the integrity of the resident operating system, recovery is initiated and one of the following actions is taken:

1. If the damage is judged to be isolated to the context of a single user, a procedure called Single User Abort is performed. In this procedure, selected areas of memory are written to secondary storage, updated file buffers are written out for the user, and the user job is eliminated. The system proceeds for all other users. The only affect for them is the brief pause to capture the dump.

2. If the damage is not isolated to the context of a single user, but certain key system tables are judged to be intact, a procedure called recovery is performed. In this procedure, selected memory data is written to secondary storage for subsequent analysis. The context for each user is then examined. All open files are closed with default options. Remaining input for batch jobs that are partially completed is discarded unless the user has specified the rerun option in his job deck, in which case the job is put back into the job queue. The accounting information is saved and the resident operating system is restored from the system device. Before resuming normal operation, accounting records are written. At this point, system operation proceeds. Terminal users must log on again. In-process transactions are automatically reprocessed.

AUTOMATIC DUMP ANALYSIS

After any recovery is performed, the monitor dump anaylsis program is initiated to aid in determining the cause of the problem. The output produced by this program consists of formatted displays of monitor and user tables, the status of the system at the time of the problem, and other data useful in problem identification.

SECURITY

CP-6 utilizes extensive security measures to prevent unauthorized use of the system. Access to the system is controlled by user authorization performed by the system manager or a designated project administrator. Memory security protects the users from the monitor and vice versa. File security prevents unauthorized access to files.

SYSTEM ACCESS SECURITY

Access to the system is controlled by user authorization, which in turn is controlled by the system manager. To access the system, the user must have a user identification. As part of the identification, the user may include a password, which is stored in scrambled form. As an added protection, identification information is not echoed at an on-line terminal during log on. A user's authorization determines which monitor services and shared processors are available to the user.

MEMORY SECURITY

Hardware protection features prevent unauthorized access to memory locations. Memory management routines clear acquired memory to prevent access to data from previous programs. A user suspected of attempting unauthorized actions may be aborted by the operator, and his or her authorization to access the system can be dynamically deleted.

FILE SECURITY

The CP-6 file system uses the four control techniques described in this subsection to prevent unauthorized access.

GRANUAL ACCESS CONTROLS

Each granule which is active in the system (except any unwritten granules of a random file) has an identification stamp in the first word of the granule. Thus, no information from any source other than the file in question is returned to the user or left in any of the user's monitor buffers unless the stamp is verified. This technique provides a high level of information security for both hardware and software error situations, and also prevents the user from reading any granule that has not been written.

FILE ACCESS CONTROL

The CP-6 file system features eight types of access controls for files (e.g., read, write, update, delete records, knowledge of its existence, access only by specified processors.) Each file may specify a combination of these accesses to be permitted to ALL, NONE, or explicit lists of accounts. In addition, a special convention permits the user to restrict access to accounts that contain specified character strings. (File control information is included within files stored on disk or on labeled tapes that have CP-6 formatted files.) Tape file management allows tapes to be semi-protected or fully protected.

DATA ACCESS CONTROLS

Data access is controlled through two mechanisms: passwords and encryption.

A user may assign a password to a file. Access to the data is denied to any user who cannot supply the password. When the password is first assigned, it is passed through a non-reversible encoding mechanism and the encoded password is entered into the File Information Table (FIT).

Encryption of data can be requested for any file organization except indexed. The algorithm is derived from the data and a seed defined by the user. This seed is not present within the file information, so even highly privileged users cannot request decryption without knowledge of the seed.

The I-D-S/II processor provides additional levels of security for I-D-S/II data bases.

Section 11

Transaction Processing

CP-6 Transaction Processing (TP) provides an on-line, interactive environment designed for high-volume, fast-response processing. CP-6 TP provides efficient data entry to and retrieval from a central data base using a variety of terminal stations that may function simultaneously.

OVERVIEW

The TP operating environment consists of software processors which combine with user application programs in both the host (the mainframe and associated file devices) and in the front-end processor (FEP). Figure 11-1 illustrates the TP operating environment.

As a fully integrated part of the operating system, TP offers the complete capabilities of the CP-6 system and a protected environment which:

- Guarantees easy installation of a TP system.
- Co-ordinates co-operating application programs.
- Minimizes use of host resources for TP.
- Assures fast, accurate data entry through the capabilities of the new Forms Processing Language (FPL).
- Uses comgroups to facilitate program development by:
  - Providing an easy-to-use READ/WRITE interface in application programs.
  - Providing a useful debugging facility. Application programs can be developed in time-sharing and batch modes and then run in TP mode.
- Provides device independence for application programs.

CO-OPERATING APPLICATION PROGRAMS

Transaction processing is performed in two separate, but co-operating application programs: a Forms Program (FP) in the FEP and a TP Application Program (TPAP) in the host. The FP interacts with an on-line terminal and its clerk/user. The TPAP interacts with a data base and creates and formats responses to requests from FPs.

Though co-operating, the FP and TPAP run independently. Execution of the TPAP is detached from events at the TP station and in the FPs.

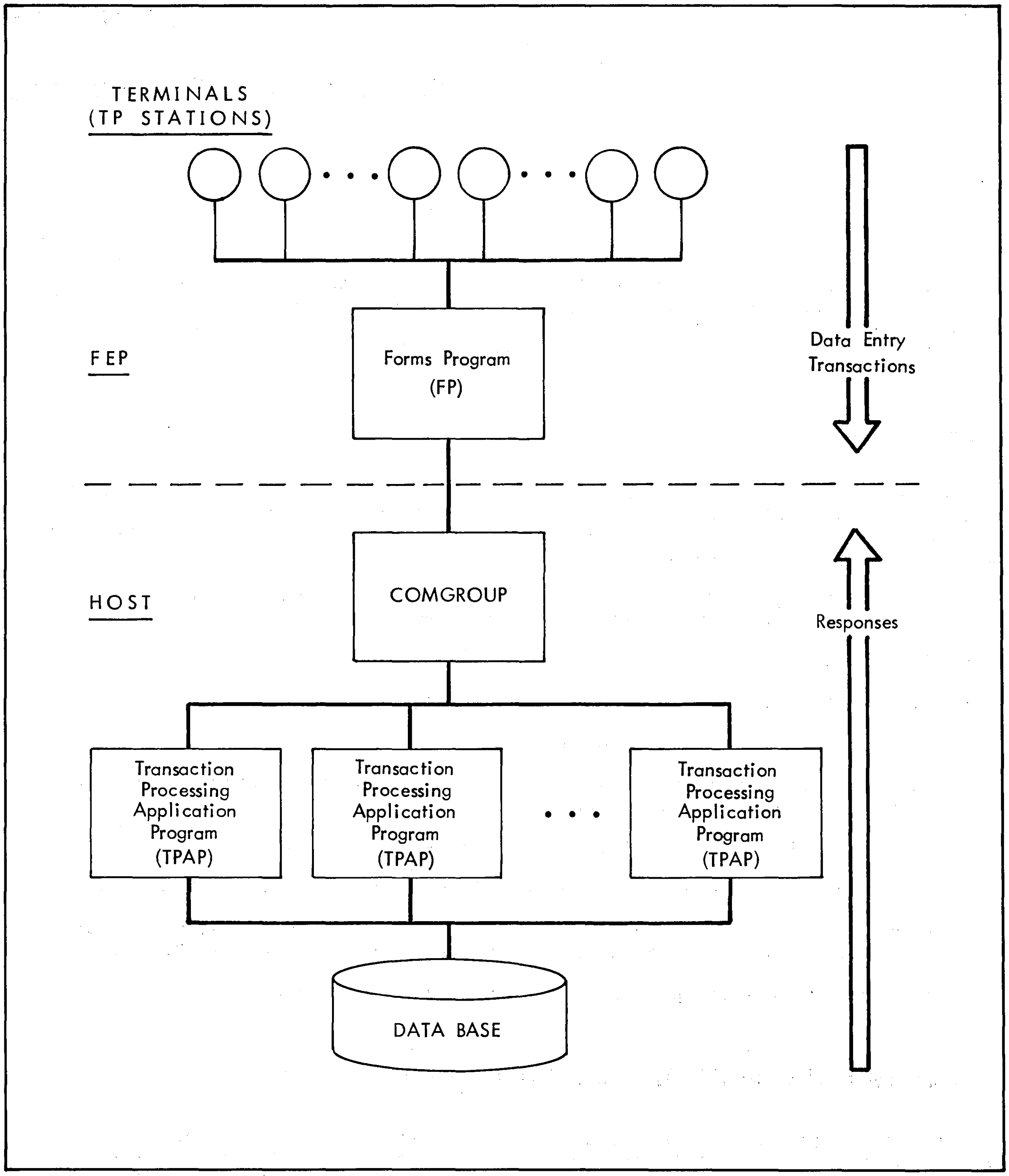

TERMINALS
(TP STATIONS)
FEP
Forms Program
(FP)
Data Entry
Transactions
HOST
COMGROUP
Responses
Transaction Processing Application Program (TPAP)
Transaction Processing Application Program (TPAP)
Transaction Processing Application Program (TPAP)
DATA BASE

Figure 11-1. The TP Environment

MINIMIZING USE OF HOST RESOURCES

CP-6 TP funnels transactions from a large number of TP stations to relatively few resources within the host. TP efficiency features in the use of host resources include:

- Rather than applying the system resources for a single 'job' to each clerk/user at a terminal, each TPAP may serve a large number of clerk/users at a variety of terminals.
- With the exception of minimal memory allocation, the resources of the host are allocated to transaction processing only for the period of time needed to process transactions that arrive in the system.
- Input transaction formatting is accomplished in the FEP.
- Separation of invocation of TPAPs from events at the TP station decreases the amount of time the TPAP need be resident in memory.

FPL

FPL is a new, flexible, powerful COBOL-like language developed by Honeywell to simplify forms processing. The benefits of using FPL in TP include:

- An easy-to-use language in which to program the essential functions of data input, validation, and automatic formatting.
- The ability to communicate with a number of stations simultaneously while sharing a single copy of procedure.
- The ability to define the interactions between terminal clerk/users and application programs.

COMGROUPS

All communication between the FP and TPAP is accomplished by means of a simple READ/WRITE interface. To receive and transmit transactions, the FP issues READ/WRITE statements, while the TPAP uses CP-6 file management READ/WRITE services.

The large volume of transactions sent from a number of TP stations is handled by the host through comgroups. Data entry transactions remain in the comgroups prior to and during processing by TPAPs. Responses written by TPAPs are stored in the comgroup, and remain in that queue until successfully transmitted to an FP.

DEVICE INDEPENDENCE

Device independence relieves the programmer of concern for device characteristics and allows a single FP to serve a range of different terminal types. A single FP can communicate with a variety of terminal types, using the same form and procedure.

The CP-6 forms processing system ensures that the various terminal features of differing types of terminals are used to the fullest extent possible. For example, for a video screen terminal, the forms processing system will adapt communication so that all constants are displayed at once, cursor repositioning occurs, and terminal features such as character validation are utilized; for an interactive teleprinter, the forms processing system sees that the constants are displayed field-by-field to suit the line-by-line operation characteristics of the teleprinter, that validation of each field is performed, and that any invalid entries are resolicited in a manner suited to the particular device.

FEATURES

The TP environment provides the structure and protection necessary for efficient on-line transaction processing. In addition, the CP-6 Transaction Processing offer these important features:

- Full CP-6 capabilities. The TP environment permits the TPAPs and TP system to make use of all available CP-6 services. For instance, file management and security is performed in the same way as in other CP-6 environments.

- Data base integrity. The I-D-S/II Data Base Manager can play a critical role in the TP environment. I-D-S/II detects and corrects deadlock on file accesses. I-D-S/II also creates transient journals and allows TPAPs to checkpoint file data for backup in case file recovery is required.

- Automatic recovery/restart. The TP system in conjunction with I-D-S/II provides an option for automatic data base roll-back and re-run of transactions which caused data base updates during system recovery (e.g., after a "soft" failure).

- Debug facilities. The sytem debugger, DELTA, is available to debug TPAPs in the TP mode as well as all other modes of access. In addition, a special test mode is available in TP for protection of the data base during debug sessions.

- Instance administration. A person designated as the administrator of an instance of TP controls operations of its application(s). The administator provides on-line control including:

  - Privacy and security. Safeguards centralized at the host provide authorization checks for TP operators and permit discretionary system surveillance.

  - Auto-logon. Special purpose TP stations such as stand-alone printers can be logged on automatically.

  - Accounting. The TP system makes special provisions to account for TPAP processing time. Options are available to account for each transaction or for a series of transactions.

  - Transaction processing timing. Options are available to control the invocation of TPAPs. For example, a TPAP can be invoked to process either one transaction at a time or a series of transactions. Transactions of a specified type may optionally be bypassed and processed later.

  - Performance evaluation and tuning. Statistics collected for each instance of TP are available to the instance administrator who may alter operating characteristics to enhance performance.

Section 12
CP-6 Time-Sharing

The CP-6 system features a terminal 'personality' that is designed to increase programmer productivity by creating a natural and powerful time-sharing environment. Access to all types of peripheral devices and rapid terminal response creates an atmosphere in which each time-sharing user appears to have the entire system dedicated to his or her tasks. Dynamic timing algorithms make the CP-6 system compatible with a wide varity of terminal types. The CP-6 system also provides utility processors that aid in accomplishing:

- Program Development.
- Program Compilation.
- Program Execution.
- Program Debugging.
- File Maintenance.
- Text Creation and Editing.

OVERVIEW

Programs to be executed via entry through the other access modes of operation may be completely or partially developed in the time-sharing mode. Since there is one common CP-6 file management facility, files used in the interactive mode are identical to those of the batch and real-time modes.

The command language (IBEX) by which the terminal user directs the course of a time-sharing session is identical to that used for a batch job (either local or remote).

Up to 500 time-sharing terminals may be simultaneously active. Under the CP-6 system, terminals can be connected to the system via dial-up communication lines or permanent circuit lines. These lines may be interfaced either locally or remotely, as described in section 9, CP-6 Communications Management.

Regardless of how a terminal is physically connected to the CP-6 system, terminal protocol is the same. After connection has been established, users identify themselves by entering log-on data: their account, their name, and (if required) a password. If the identification is valid and consistent with information maintained by the monitor, the user's on-line session is initiated and the system prompts the user for commands. If the identification is invalid, the CP-6 system sends an error message and requests the user to resupply the log-on data.

An on-line session is terminated by entering a simple command to log off. The CP-6 system then transmits selected accounting information and offers the user the opportunity to log on again. Thus, separate accounting for separate functions may be achieved by a change of account number and/or name.

TIME-SHARING FEATURES

The CP-6 system has an extensive set of commands which enable the terminal user to edit terminal input, control terminal output, and control the course of program execution. The most widely used features are summarized below.

TYPEAHEAD

CP-6 terminal I/O routines allow terminal users to type input during the time computation is taking place or output is being typed at the terminal. Such input is not echoed to the terminal immediately. Instead, the data is stored until the proper time for its utilization; that is, the time following programmatic requests for input. Thus, in the script typed at the terminal, input appears following the query asking for it even if the input was typed sometime previously.

TERMINAL PROFILES

CP-6 terminal profiles identify terminal types. Profiles contain a detailed description of terminal characteristics, such as character code set, timing information, a list of the terminal's features, and how to perform cursor positioning. Profiles may be defined by the installation, using SUPER. Definition files for most terminals normally used with CP-6 are included in the installation materials provided by Honeywell.

The user may specify the name of the profile during or after logging on, or may have a specific profile invoked automatically during log-on.

The CP-6 system provides character code conversion for the ASCII character set, some variations of ASCII, and bit-paired and typewriter-paired ASCII APL. The system can automatically adjust to the ASCII or ASCII APL character sets during the connection process. The character set may also be changed during the session via the ESC $ escape sequence, a PROFILE command or programmatically.

OUTPUT EFFICIENCY

Positioning functions directed to the terminals are optimized. The effects of space, backspace, horizontal tab, carriage return, and line feed characters (and combinations of these characters) are analyzed before sending positioning information to the terminal. Then the optimal combination of the carriage return, line feed, space, backspace, direct cursor position, absolute tab, cursor forward, cursor back, cursor up, and cursor down functions is sent to the terminal. This optimization process can result in greatly increased terminal throughput. This techinque also allows any program to take advantage of advanced device features without knowing what device is being used, thereby providing a high degree of device independence.

Several timing algorithms are supplied to determine when timing fill characters are sent to the terminal. These characters provide delay for positioning of the carriage before and/or after positioning functions (e.g., carriage return, line feed, absolute tab), and after erase and control functions. The algorithms are further tailored to the device by way of timing parameters specified in the profile, yielding optimal output efficiency.

The CP-6 system supports flow control for buffered devices using the DC1/DC3 (XON/XOFF), ETX/ACK, and CTS (Clear To Send) disciplines. This support means data can be sent to the device at the maximum rate the device can accept data.

TRANSPARENT MODE

A program may request that no translation or other processing be performed by CP-6 terminal I/O. In this case, characters are moved between program and terminal without change in the bit structure. This mode is useful for connecting to special equipment, tape cassettes, plotters, paper tape, or other computers.

PAGINATION AND FORMATTING

The CP-6 system offers two types of pagination: logical and device.

Logical pagination is similar to line printer pagination. System service calls, IBEX commands, or a line printer compatible forms description enable the user to establish page parameters, including page width and length, top and bottom margins, and the channel number to line number correspondence for skip to channel operations. When the end of the logical page is reached, a page break is performed, and a heading is issued. Headings generated include user specified text and reflect any user-supplied indentation, page numbering, and space-after parameter values. Input and output lines are folded to fit within the page width. All VFC (Vertical Format Control) codes legal for a line printer are also handled for terminals. Typing ESC L causes a page break. Typing ESC - signifies the terminal is positioned at the perforation. Simulated perforations may be printed on non-perforated paper to ease fanfolding or cutting at page boundaries.

Device pagination allows the operator of a screen terminal to view output before it is scrolled off or over-written. Output is halted after each screenful of data. The operator then types a carriage return to continue. The relative pagination mode computes the page position relative to the last read, eliminating page halts when the user is interactive and not generating full pages of contiguous output. Device pagination may be turned on or off at any time by typing ESC A .

TERMINAL INPUT FUNCTIONS

The CP-6 system supports a number of terminal input functions that allow the user great flexibility in preparing terminal input. Terminal input functions define special characters and escape sequences that allow the user to edit terminal input, control input conversion, control tab simulation and perform miscellaneous functions. Table 12-1 lists the CP-6 terminal input functions.

EDITING TERMINAL INPUT

Visual fidelity is maintained on most screen terminals. When character insertion, replacement, and deletion is performed, the image on the screen is constantly updated to the current contents of the input record. As this update is performed in place, a minimum of space is consumed on the screen, thereby minimizing the scrolling off of important information. Since characters cannot be erased on hardcopy terminals, editing causes characters to be printed that note the updates to the record. However, all editing functions work on both screen and hardcopy terminals.

Terminal editing may also be used on program-supplied data. The user program may pass a record to the CP-6 system, allow the operator to edit the data, and send it back to the user program. The CP-6 EDIT and APL processors use this feature to provide terminal editing for data in their files.

CONTROLS OVER INPUT CONVERSION

Character conversions are generally done automatically by the CP-6 system in the proper way for each terminal type. However, the input conversion functions allow the user to control upper and lower case conversions.

TAB SIMULATION

The tab simulation functions control the effect of simulated tab stops on the terminal and on the program.

The user may also establish the setting of the simulated tab stops using either programmatic or execution control commands.

USER INPUT FUNCTIONS

Users may define their own input functions by using IMP. Input functions define special characters and escape sequences that the CP-6 system is to interpret. The interpretation yields character strings to be processed as input text and/or system special characters and escape sequences, or as output text. For example, the following IMP command defines an input function that generates a tab setting command when a # character is entered by the user:

```
    ADD SPECIAL_CHARACTER '#'
      TEXT='TABS 10,20,30,40,50'
      READ  INPUT_FUNCTION ECHO
```

IMP may be used to re-define keys on a keyboard, define function keys, and define escape sequences and special characters to invoke commonly used keystrokes.

Table 12-1. CP-6 Terminal Input Functions

Terminal Function Category	Input Function Command	Description
Editing Terminal Input	BS	Moves the cursor left one column (backspace).
	DC2	Moves the cursor right one column (control-R).
	DEL	Deletes either the last character typed or the character at the cursor position.
	ESC CR	Positions to the beginning of the input record.
	ESC D	Recalls the last input record as if it has just been re-typed. It may then be edited and re-submitted.
	ESC J	Enables editing of the current line by insertion of characters.
	ESC K	Deletes all characters from the cursor to the end of the input record.
	ESC M	Resets the logical overstriking mode. Typing on top of existing characters causes the new character to replace the old character(s).
	ESC N	Positions to the end of the input record.
	ESC O	Sets the logical overstriking mode. Typing on top of existing characters causes overstrikes containing both the old and new characters (separated by a backspace) to be formed. This function is typically used for underscoring.
	ESC R	Retypes the current input record.
	ESC V char	Positions to the specified character.
	ESC X	Deletes the current input record (usually a line).
	HT	Positions to the next input tab stop (horizontal tab).
Input Conversion	ESC U	Reverses the setting of a mode in which received lower case letters are converted to upper case.
	ESC)	Causes subsequently received upper case characters received to be converted to lower case. Thus lower case characters may be input on an upper-case-only terminal.
	ESC (	Removes the effect of ESC) .
Tab Simulation	ESC C	Reverses the setting of tab relative mode (ON to OFF or OFF to ON). If this mode is ON, input tab stops are assumed to be relative to the end of the prompt.
	ESC S	Reverses the setting of the space insertion mode. If this mode is ON, spaces are delivered to the reading program when tab characters are received.

Table 12-1. CP-6 Terminal Input Functions (cont)

Terminal Function Category	Input Function Command	Description
	ESC T	Reverses the setting of tab simulation mode. If the mode is ON, spaces or other positional characters are sent to properly position terminal output when a tab character is received from either the terminal or a program.
Miscel- laneous	CONTROL-Y	Returns control to the command processor and deletes queued input and output.
	CONTROL-X	Deletes queued input and output.
	ESC B	Gives control to the interrupt point of a program associated with the terminal, and deletes queued input and output.
	ESC E	Toggles echoplex mode. Echoplex may be turned off to prevent the printing of sensitive information (such as passwords) entered from the terminal.
	ESC F	Signals an end of file.
	ESC H	Temporarily halts terminal output.
	ESC I	Simulates the entry of an HT (horizontal tab).
	ESC Q	Requests an acknowledgment that the communications processor and host are alive, and obtains the current host scheduler state.
	ESC W	Deletes all output until another ESC W is typed or until a read is issued to the terminal that is not satisified by typeahead.
	ESC Y ESC ESC	Returns control to the command processor, and deletes queued input.

ENTRY OF JOBS TO THE BATCH QUEUE

When the on-line user does not wish to sit at the terminal and attend the execution of a long process or wait for resources such as tapes, the terminal batch entry facility can be employed. This facility allows the user to enter a job into the batch job queue for execution in the batch processing mode. The user can then disconnect from the system or start another time-sharing task.

This service allows time-sharing users to create and edit a control command file which will direct the execution of their jobs. At any time after submitting a job control file, the user can request the status of the job.

CP-6 responds by telling the user that either:

- o the job is enqueued. The number of jobs ahead of his in the queue is displayed.
- o the job is running.
- o the job is completed.

The user can also cancel the job from the on-line terminal. After the job has completed, the user can examine files created by the job.

Even if the batch mode is not operating concurrently with the time-sharing mode, jobs may be entered into the batch job queue for subsequent execution as soon as the batch mode is activated by the operations staff.

COMMUNICATION WITH OPERATOR

Communication of control instructions to the CP-6 system is accomplished through the IBEX processor. Since the on-line user is in direct control of the computing tasks, the need for the vast majority of special instructions to the computer operator is eliminated. However, the need for some communication between the on-line user and the central operator still exists (for example, to request the mounting of tapes and disks or to request information).

CP-6 provides facilities for the on-line user to transmit messages to the central site operator. When the message appears on the operator's console, the transmitting terminal and account is identified with the incoming message. The central operator can then carry on a dialog with the individual on-line user.

For users not currently logged on, the central operator may issue a 'greeting message'. This message is stored by the CP-6 system and is presented to the user during the log-on process.

AUTOMATIC PROCESSOR ASSOCATION

The time-sharing mode allows the user to work at the terminal, interacting 'directly' with a CP-6 processor or with a user-written program, through monitor routines that do not make themselves apparent to the user but which facilitate the interaction taking place.

In general, a time-sharing user may interact with a varity of processors during an on-line session. However, the system manager can restrict a user to one selected processor. This feature is valuable when a user who is unfamiliar with CP-6 is being introduced to the system or when a particular user requires only limited services.

FILES OF TERMINAL COMMANDS

The CP-6 user may prepare a series of command in a file and later call for their execution with a single command (XEQ). At the time each such file is invoked for execution, field and string substitutions may be performed before the execution begins. Execute files may contain any level of nesting.

AUTOMATIC SAVE FOR DISCONNECT

CP-6 optionally preserves a user's program when a line disconnect occurs before the user has logged off, and provides a method of reconnection of the preserved program when the user calls back. Files remain open and properly positioned so that the program can be continued as if an interrupt has not occurred.

When a line disconnect occurs, the suspended program image is retained for a fixed length of time. The retention period is established as a system parameter and may be modified by the operator at any time. (The operator can also abort a user when the user is in the suspended state.)

When the disconnected user logs back on the system, the system recognizes that a program image exists for the account/name combination and issues the following message:

```
PROGRAM HELD.  RECONNECT(Y/N/D)?
```

The user then responds with either Y, N, or D. If Y is specified, the user is reconnected to the suspended image and continues from the point of disconnect. If D is specified, the user is not reconnected, and the suspended image is discarded. If N is specified, the user is not reconnected, but the suspended image is saved.

SPEED AND FORMAT DETECTION

CP-6 detects the speed and format of calling terminals by examining characters typed by the user and sets the hardware line interface module appropriately. Speed detection is supported at 110, 150, 200, 300, 600, 1200, 1800, 2400, 4800, 9600, and 19,200 baud. The character set (ASCII, or bit-paired or typewriter-paired ASCII APL) and the parity (even, odd, space or mark) is also sensed and adapted to. These features eliminate the need to segregate lines by speed or format and result in a lower per-line cost.

TERMINAL TAPE INPUT

CP-6 provides facilities for terminal input from either paper tape, cassette, or other compatible sources. Tape may be prepared on off-line terminals and subsequently read on-line after the user has logged on and a prompt for data on the tape has been issued.

Section 13

CP-6 Batch Processing

The CP-6 system offers full-service, multiprogrammed batch facilities, ideal for the production data processing environment. Batch jobs can be conveniently submitted from local card readers, remote sites, time-sharing terminals or from running batch jobs. Since the execution control language is identical to that used in time-sharing, batch facilities are made available to a larger class of user. The CP-6 philosophy of commonality among the basic aspects of the system allows the user to construct interdependent program systems, in which several programs work together to accomplish a single task. In addition, a scanning feature is available to check execution control syntax prior to execution, thus significantly reducing total debug time.

OVERVIEW

In the CP-6 system, multiprogramming (the concurrent operation of several jobs) significantly extends the throughput of the computer system because the system is able to achieve several levels of overlap in the processing of jobs. I/O of one job is overlapped with computation of another, and I/O of several jobs is overlapped through the effective use of the hardware's multiple channels to I/O devices. This overlap applies not only to jobs within the batch mode but also to all other jobs. Thus use of CPU, disks, tapes, and other system resources is maximized. As described in Section 14, Remote Processing, a CP-6 system may itself act as remote workstations. Jobs may also be received from any systems that support the HASP transmission protocol. Commands for jobs, regardless of source, are stored temporarily while awaiting execution in spooling files as described below.

RESOURCE CONTROL AND SCHEDULING

Batch jobs are protected from execution-time resource conflicts or deadlock situations by the CP-6 job scheduling algorithms. Each batch job contains a command which specifies the required resources and the system ensures availablity of all required resources before starting the job. In contrast, time-sharing jobs may request resources dynamically, but must be prepared to respond to a 'not available' message.

The CP-6 system can concurrently run a large number of batch jobs. Each job is run in a batch job class. (Batch job classes are described further in Section 17.) The system manager can designate up to 16 batch job classes. In this way, up to 500 jobs can be run concurrently, providing full resource use and adaptability to such circumstances as evening, night, and weekend shifts.

Each of the batch job classes has a defined resource profile that the system manager can change dynamically. The profile describes the ranges of job resources which are allowed when running in each class, and the number of batch jobs that can run concurrently in each class. In order to qualify for a given class, the resource requirements of a job must match the class profile. A job may fit in many classes.

Whenever a job class is available and an event (such as job termination, resource released, or job submitted) occurs, the batch job scheduler selects the first (highest priority) job with requirements matching the class profile. Both the job and its required resources are assigned to the job class and the job begins execution.

CP-6 optional job scheduling controls allow the job submitter to dictate additional scheduling criteria to the system's batch job scheduler. Unlike global controls, which are established by the system manager via the CONTROL processor, these scheduling criteria are local to the submitter's authorized account. They are specified on the job's control commands via the DEFER, FOLLOW and ORDER options. The DEFER option defers scheduling of the job until a particular date and time. The FOLLOW option orders execution by job name; subsequent jobs submitted can run only after the completion of specific, named jobs. The ORDER option indicates that execution of the job must follow completion of any previously submitted job. Thus a series of jobs, each using results from the previous job, can be submitted at once, but are forced to execute serially.

WORKSTATIONS

The CP-6 workstation concept organizes the devices connected to the system into manageable groups. Workstations may be remote or local to the central site, and usually include unit record peripherals and tape devices.

The CP-6 system manager defines workstations tailored to the installation's requirements. For example, one line printer in a university environment may be defined via a workstation restricted to administrative work, while a second line printer intended for student use can be located in a less secure work area. The CP-6 system then automatically routes the administrative or student output to the appropriate workstation. An operator's console may be placed at each workstation that is defined to send or receive only the messages that are pertinent to the individual workstation.

A workstation name can also be used to address any device attached to the CP-6 system. For example, a line printer at a workstation named BOSTON is addressed as LP@BOSTON. All CP-6 users, however, are assigned a workstation of origin and need only specify a device's generic name for the I/O to take place via the assigned workstation.

CATALOGED PROCEDURES

The CP-6 system allows the programmer to create a file of execution control commands and call for its execution as a job or portion of a job. This facility is available to both batch and on-line CP-6 jobs. As the execution of a cataloged command file is requested, parameters may be substituted in previously defined fields of the file. A file may, within itself, call for execution of other files, and programs may create files which are inserted in the on-going command stream.

SPOOLING

A spooling facility is provided to help eliminate bottlenecks associated with slow speed peripheral devices. This spooling facility consists of monitor routines that transfer information between secondary (disk) storage and unit record peripheral devices concurrently with jobs being run. To transfer information between a user's program and this secondary storage, a 'cooperative' monitor routine is used.

The spooling system performs complete buffering between I/O devices and the user's program. Also, the current job may be running while the output of a previous job and the job file for a subsequent job are being handled by spooling. The CP-6 spooling system is depicted in Figure 13-1.

Spooling files are normally written to disk before being output. However, for certain long-running data processing programs, the 'concurrent output mode' is provided. When in this mode, the spooling system begins printing a program's output before the program has completed. Thus a 14-minute job which produces 14 minutes of printing uniformly over the execution time of the job will complete printing in about 14 minutes rather than 28 minutes.

Spooling files are standard CP-6 files and therefore may be moved individually or collectively to tape or pack for removal from the system as required in the day-to-day management of the installation.

Normally, one operator's console is defined to have control over all spooling files in the system. However, a console assigned to a particular workstation may be used to control only those files associated with its assigned workstation.

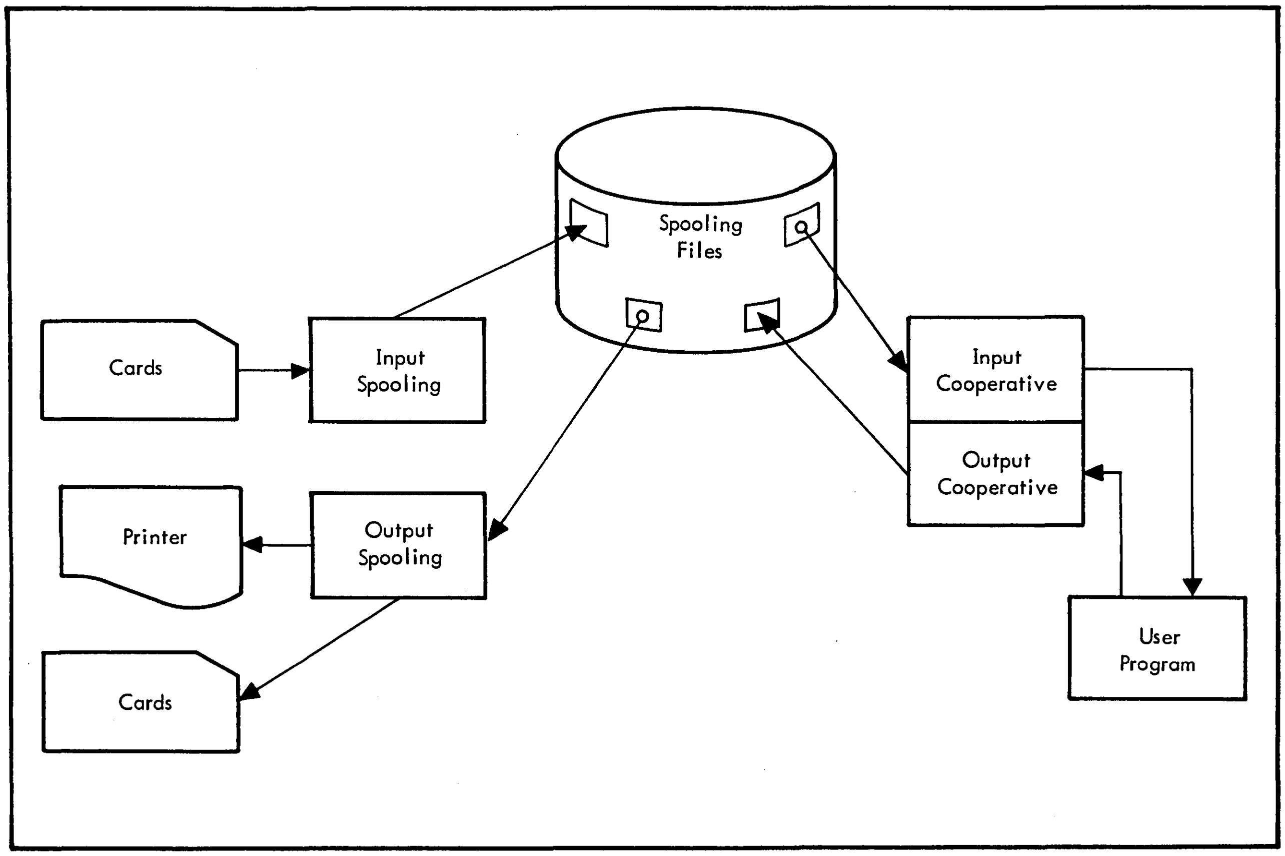


Figure 13-1. CP-6 Spooling

Section 14

CP-6 Remote Processing

CP-6 remote processing brings the facilities of a central CP-6 site to the actual work locations where that power is required. The extensive CP-6 communications features (described in Section 9) work together with the CP-6 remote processing facilities to provide a flexible and extensive communications environment.

OVERVIEW

Important features of the remote processing system include:

- A remote site can be another large-scale computer, and files of data may be transferred between user programs at the central and remote computers.
- Through monitor and user interfaces, virtually any type of device (e.g., tape, disk, and plotter) can be accessed at a remote terminal.
- As illustrated in Figure 14-1, any user of a CP-6 system can communicate with a variety of devices at one or several remote sites.
- As illustrated in Figure 14-2, a CP-6 system may act as the central site to some remote terminals and as a remote terminal to other computers simultaneously.
- Workstations need not be remote nor are they necessarily composed of physically associated devices.
- Definitions of workstations can be added, deleted, or modified during system operation.

THE ENVIRONMENT

CP-6 remote processing accesses remote facilities as sets of independent devices indistinguishable from local devices, and defines workstations for functional purposes independent of physical relationships. These access and definition techniques provide both a flexible means of accessing remote facilities, and a consistent, configuration-independent means of accessing all devices.

Access to remote facilities is provided through the standard CP-6 services for managing logical devices and spooling. All details of both communications line management and multiplexing of several apparently independent devices to a single channel are transparent to the user of remote facilities.

Default device selection, including the allocation of an output device for a spooling file, is made from the device of the appropriate type at a particular workstation, rather than from the whole system. Each job has an associated 'workstation of origin', convenient to the owner of the job.

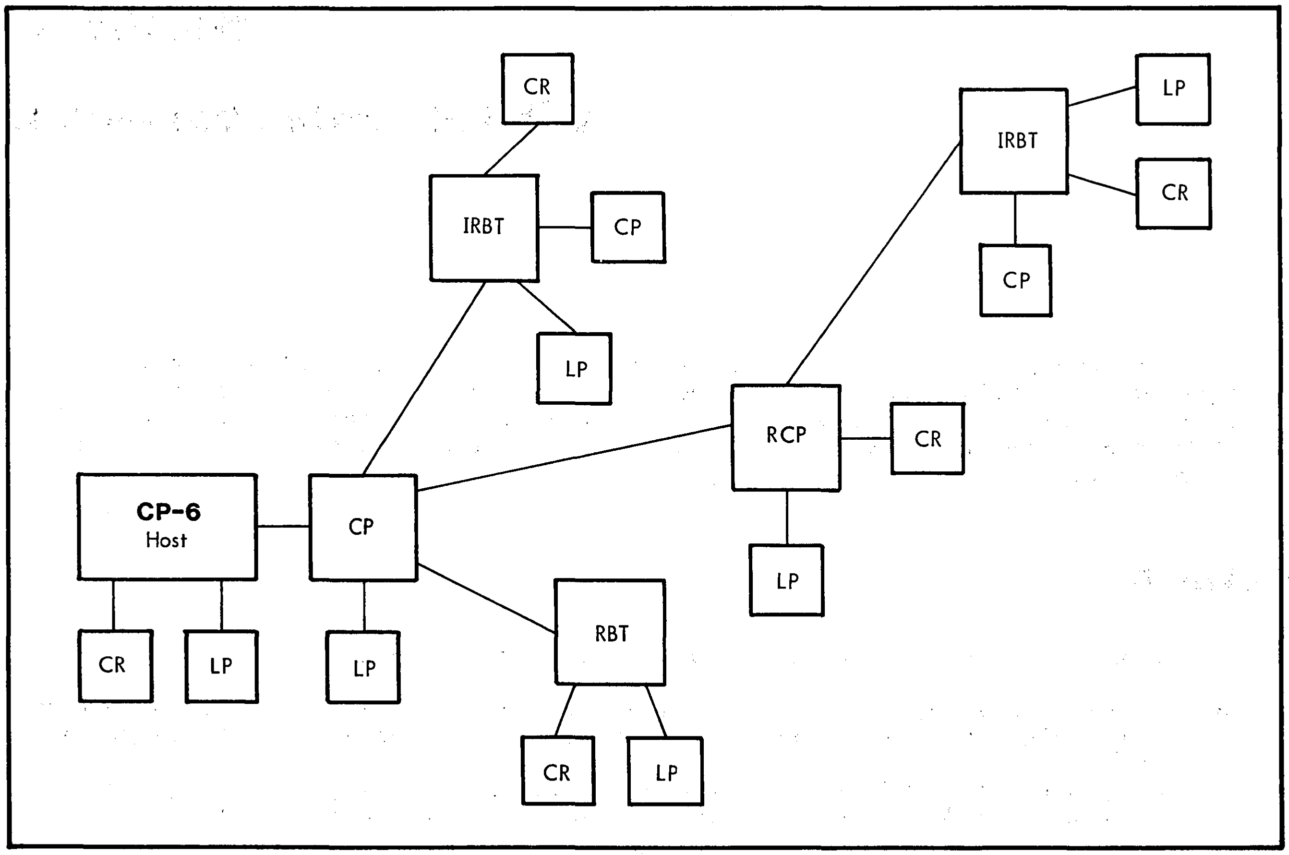


Figure 14-1. CP-6 Remote Processing Example

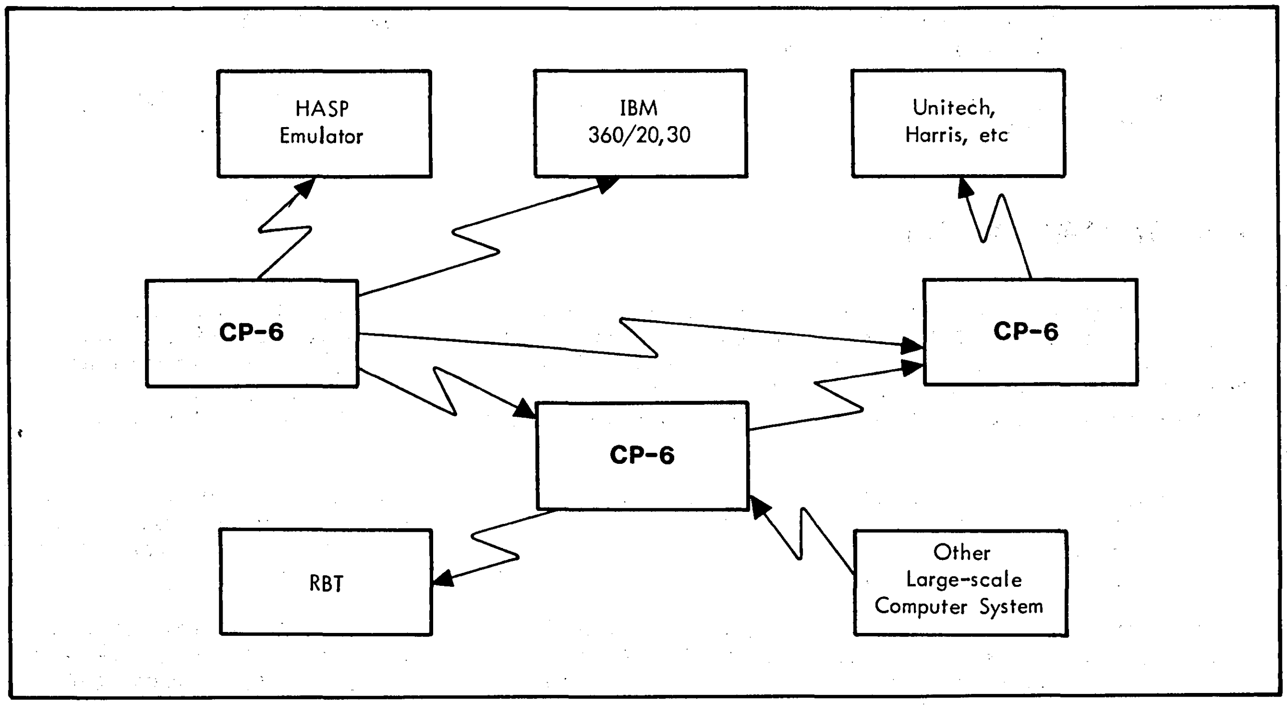


Figure 14-2. CP-6 Remote Processing Communications

A workstation can be defined independently of the physical connection of devices; that is, a workstation may include all or only some of the devices at several different locations. Additionally, devices can belong to more than one workstation, providing more flexible grouping of devices for device selection, access restriction, and assignment of operator consoles.

USER INTERFACE

CP-6 users access output devices at remote processing terminals by creating spooling output files that are destined to be sent to a particular remote device. The remote device may be:

- Selected by default. If not specified, the device will default to a device of the specified type via a workstation established for the user.
- Explicitly specified. The user may specify both the device type and workstation.
- Defined generally through a logical device. The user defines logical devices through the LDEV command and the M$LDEV monitor service.

Remote devices are accessed in the same way as local output devices receiving spooling output.

CP-6 users transmit data from remote sites to the CP-6 system as streams of consecutive records, which upon receipt at the CP-6 site are collected into files. Such files can be 'job' files to be scheduled for batch processing, or other files available directly to the user. Remote input capabilities are equivalent to those available from local card readers.

TERMINAL SUPPORT

CP-6 remote processing supports the HASP II version 4 multileaving protocol in non-transparent EBCDIC mode, and the IBM 3780 BSC protocol in non-transparent EBCDIC mode with certain options. CP-6 will act as the master or the slave station for the HASP protocol, and as the master only for 2780 and 3780 protocols. Support of these protocols allows the following interaction with remote facilities:

- Devices can be accessed at, and jobs may be submitted from IBM 2780 and 3780 terminals, and terminals that emulate them.
- Devices can be accessed at, jobs may be received from, and sequential data may be exchanged with Xerox 530 XSP system.
- Devices may be accessed at, and sequential data and jobs may be exchanged between CP-V and CP-6 systems.
- Streams and jobs may be exchanged with systems supporting HASP terminals.
- Devices may be accessed at, and jobs may be submitted from a HASP terminal.

INTER-SYSTEM FILE TRANSFER

When the remote site is another CP-6 system, Peripheral Conversion Language (PCL) commands permit transfer of CP-6 managed files between machines. The on-line or batch user may copy, create, list or delete files in another CP-6 system via the CP-6 communications network.

To use these special features of PCL, the user must be authorized at the remote site as well as the local site.

A file may be renamed as defined in the COPY command. Appropriate messages report errors, security violation attempts, and successful file transfers to the users.

CP-6 Distributed Real-Time Processing

The real-time processing facilities of the CP-6 system provide significant enhancements over competitive systems. Real-time is an important access method in the CP-6 multi-use environment, and is particularly powerful when utilizing the filing, data base, and reporting capabilities of the CP-6 system. The processing load is distributed over several local and remote computers that work in coordination with the host computer.

OVERVIEW

All hardware connection to real-time external devices is provided through separate real-time computers. These processors carry the parts of the real-time task which are directly associated with data acquisition or process control. The host system provides a high performance engine for the computational and data base portions of the real-time task.

Figure 15-1 shows a CP-6 distributed real-time system, illustrating some of the possible hardware connections. A single CP-6 host processor is connected to five real-time and communications processors. Local processors for real-time and communications, labeled RP and CP respectively, are connected to the host through a special coupler. This coupler provides one megabyte half-duplex data transfer over a 75-foot multiwire cable. Remote real-time and communications processors, labeled RRP and RCP respectively, are connected over communications facilities, either dedicated or switched. Individual connections are limited by microcode processing time to 72K bytes/second (and lower figures for complex line disciplines). However, multiple lines may connect processors for both increased bandwidth and reliability.

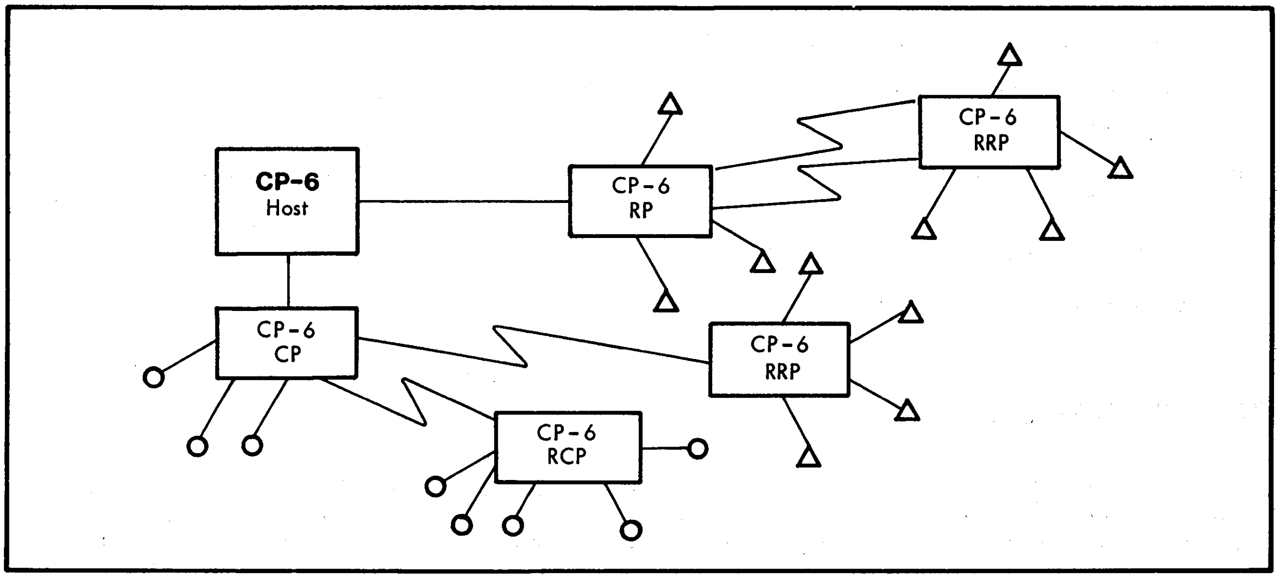


Figure 15-1. CP-6 Distributed Real-Time System

Software in the host includes the less time-critical real-time programs of the various applications. Software in the RP consists of the more time-critical real-time application programs plus the service routines of the real-time system for interprogram communication, connection to interrupts and I/O gear, timer services, and program loading and startup.

REAL-TIME SOFTWARE DEVELOPMENT

The CP-6 system includes a collection of programs that provide preparation and check-out of user real-time programs designed to execute on the RPs and RRPs of the CP-6 system. Compilers, assemblers, and associated programs are provided for program preparation. Facilities are provided in both the host and the RP for program debugging.

REAL-TIME SERVICES

All real-time services are entered through a common system interface. The hardware protects programs from each other and prevents programs from damaging the operating system. Those services include:

- Intertask signalling (within and between computers).
- Intertask message passing (within and between computers).
- Type and priority searching of message queues.
- Connection to one of 60 hardware interrupts.
- Priority scheduling of tasks.
- User control of hardware traps.
- Task creation, loading, initialization and termination.
- Clock service, measuring both program execution time and elapsed time. (Several timing operations may run concurrently.)
- Periodic task initiation.
- Memory and buffer management.
- Read/write interface to supported peripheral devices and digital I/O devices.
- User entries on system initialization, recovery, crash, and power failsafe.
- Memory sharing between tasks.
- Management of system resources.

PERFORMANCE

Response to interrupt signals in the real-time computer (i.e., the time from signal until entry to the designated highest priority program) is in the under 1 millisecond range, including context switching time and monitor overhead.

Response in the host is in the under 10 millisecond range, including all context switching and monitor overhead. This response is measured from the acknowledgment by the CP-6 system of an interrupt signalling the arrival of a message from a real-time program until the time the designated host program is entered.

Section 16

CP-6 System Programming

The CP-6 system provides a set of convenient tools designed to aid in the development and maintenance of shared entities, thus allowing the system manager to tailor the system to the installation needs. Since a large portion of the operating system is coded in PL-6, system programming can be accomplished with speed and efficiency. These features result in more effective support from a smaller support staff.

SHARED ENTITIES

The CP-6 system recognizes five types of shared entities:

- Shared libraries
- Shared (language) processors
- Command processors
- Alternate shared libraries
- Shared debuggers.

Shared libraries reside in the user's working space at a fixed origin within the Instruction Segment Register (ISR) segment (see Figure 16-1) and are shared via the page table. Library data is always at a second fixed origin within the ISR segment. Standard linkage to shared libraries utilizes the direct subroutine branch (TSX) instruction.

Shared processors that run as user programs in the user's working space utilizing dynamic data and user data segment space as required are shared via the page table. User programs may be shared in this way as an installation option.

All command processors run in one working space which consists of procedure and constants plus the page table for the working space. They obtain data space from the user's working space data segments that are reserved for command processors. Only one command processor may be associated with a given user at a given point in time; therefore, the linkage segment for the command processor is carried in the user working space. There is no direct linkage (CALL-RETURN) between command processors and user programs; instead, the interface is in the monitor.

All alternate shared libraries (ASL) run in one working space which consists of procedure and constants plus the page table for the working space. Like command processors, ASL obtain data space from those user's working space data segments that are reserved for ASL use. The interface from the user to the ASL is through the CALL form of CLIMB, while the ASL itself uses the RET form of CLIMB to go back to the user program.

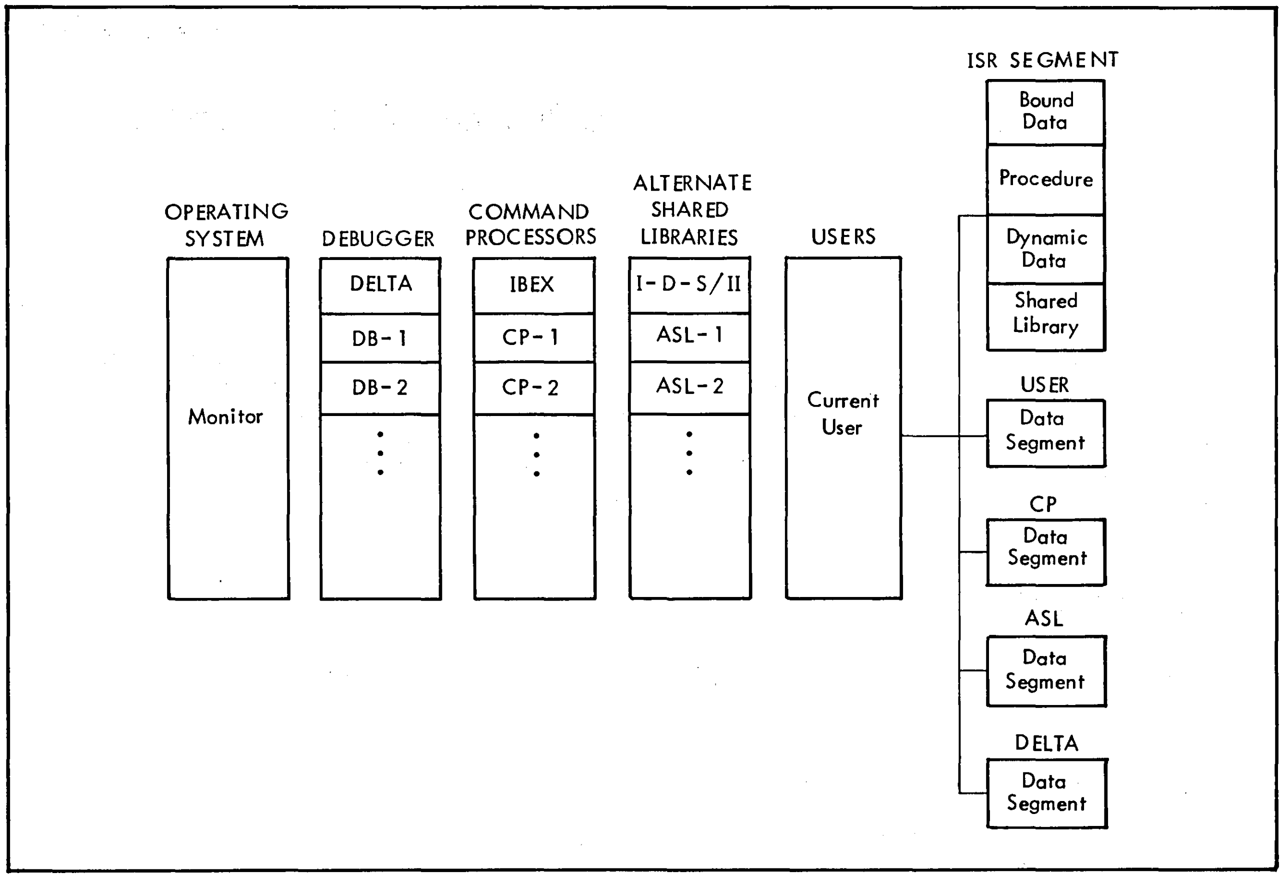


Figure 16-1. CP-6 Context for Sharing

The DELTA debugger's procedure resides in its own working space and does not occupy any of the user's virtual memory. In addition, DELTA needs its own data area within the user's working space. Descriptors in DELTA's linkage segment provide full access to all segments within the user's working space, as well as access to DELTA's procedure in its own working space and to the debugger data area. DELTA does not have access to any other special shared processor data area or procedure. DELTA is not entered directly by users; rather the monitor enters DELTA for the user either as the result of a return to IBEX followed by a DELTA command or on certain faults. (DELTA plants fault-causing instructions at breakpoint locations.) Users may also use the RUM mode of DELTA to make permanent patches to run unit files. Privileged users may use the ANLZ mode of DELTA to examine/change the running monitor, or to examine system dump files.

INTERFACES

In the CP-6 system, all programs interfaces -- both intra- and inter-language -- are defined via a set of standard calling sequences, which, for generality, are defined at the assembly language level. All compilers adhere to these calling sequences in the code that they generate. One of the standard calling sequences is the set of monitor services calls. Part of the calling sequence specifies the location of a Function Parameter Table (FPT) that contains information pertinent to the service request, as well as information pertinent to the returning of any result.

EXCEPTIONAL CONDITION HANDLING

Facilities are provided whereby programs can specify procedures that are invoked on occurrence of exceptional conditions such as traps, break control from a terminal, elapsed time run out, and event posting. Information relating to the nature of the condition is conveyed in a frame on a stack devoted to exceptional condition handling, thus permitting processing of multiple simultaneous conditions or recursion on a given condition. Exit from condition handling may be of three types: resume processing from where the trap occurred, transfer to a previously specified label (unwinding automatic storage in the process), or merely pop the frame on the condition stack. Debugging of exceptional condition handling is completely supported by DELTA.

ACCESSING SHARED ENTITIES

Centralized enqueue/dequeue facilities are provided for general use in controlling access to any phenomena (data, files, programs, etc.) that can be shared by more than one program.

INITIALIZATION AND DATA

Of the five shared entities discussed above, only shared libraries and shared processors have any data space in the Instruction Segment with which to work when they are first entered. Command processors, debuggers, and alternate shared libraries may have initialized data segments that are acquired at initial association time.

STAR FILES

The names of one type of CP-6 file are guaranteed by the system to be unique for a given batch job (or terminal user): the star file. A file is considered a star file if, and only if, the first character of its name is an asterisk (*). The directory for star files is named *T and is itself cataloged in the Job Information Table (JIT). Star files exist only as long as the job (or terminal session) exists, and they never appear in any of the system catalogs. The following star files names are reserved:

Star File Name	Contents/Purpose
*A	Assign/merge information
*G	Object unit output from compilers
*I	Reserved for I-D-S/II
*L	Default run unit
*N	Used by LDLNK
*S	Step accounting, miscellaneous IBEX information
*X	Reserved for the monitor.

All other names are available (e.g., for use as temporary files) and are not restricted to two characters.

STANDARD DCBs

By convention, all processors are expected to use standard Data Control Blocks (DCBs) for standard I/O functions. For example, source input is read through the M$SI DCB. The important standard DCBs are:

DCB	I/O Functions
M$SI	Source input
M$UI	Update input
M$DO	Diagnostic output
M$LO	Listing output
M$SO	Source output
M$OU	Object unit output
M$ME	Command stream input

Section 17

CP-6 System Managment

The system management facilities of CP-6 are far more powerful than those of competitive systems. Through a complete package of system management processors, a CP-6 system and its associated resources can be dynamically tuned for maximum performance and cost efficiency. In addition, the RATES processor allows the system manager to define monetary charges for any level of activity.

SYSTEM DEFINITION

CP-6 system definition facilities include the program and procedures in the system which accomplish configuration and reconfiguration. Some of this process occurs during cold start, parts during warm start (or recovery), and parts can be dynamically changed during system operation.

The startup process takes about 45 minutes and does not require user interaction or the presence of specially trained personnel.

The start-up adaptation of the CP-6 system to its hardware includes generation of tables for peripheral devices, I/O enqueuing, physical memory management, and managing the CP-6 users. Certain tables grow and shrink dynamically, adapting to the demands of load during system operation. This procedure relieves the system manager or analyst of the difficult chore of 'pre-guessing' requirements. Tables handled in this manner are:

- File access control tables (CFU).
- Enqueue/dequeue tables.
- I/O accelerator tables and buffers.
- Communication context (line) tables.

Network definition does not require a startup procedure. CP-6 carries definitions of remote processing workstations in files that may be created and altered during system operation.

SYSTEM PERFORMANCE CONTROL

The system manager can allocate the resources of the system to jobs with certain attributes by defining a set of batch job classes under which diverse categories of jobs may run. A batch job class is characterized by a set of job attributes. Physical system resources such as memory, spindles, or tape drives are not permanently allocated to a particular batch class. All jobs executing in the various batch classes draw their physical resource requirements from a common pool without regard to the class under which they qualified for execution, except that the numeric limits that pertain to that class will apply. Examples of attributes that comprise a batch job class profile are:

- Minimum and maximum job execution time.
- Minimum and maximum amount of main memory.
- Minimum and maximum number of disk drives.
- Minimum and maximum number of tape drives.

All jobs submitted for CP-6 batch execution share the same input queue (the batch job queue). Jobs are selected from this queue for execution in the batch job classes.

Scheduling is performed in the following manner:

1. Available resources are determined.
2. The highest priority job requiring only available resources is selected.
3. The batch job class tables are searched for a job class that fits the requested resources and is currently available.
4. If no job class is available for the selected job, the next job is considered as in step 1, 2, 3.

In summary, batch job class definitions are a primary factor in the job selection process. The system manager may direct processing of any particular category of jobs by means of those definitions.

In a time-sharing/batch processing system, emphasis may be given to batch processing by opening up more batch job classes. The CP-6 system is queue-driven; tasks are selected from prioritized queues without regard to the source of the request (i.e., on-line, batch, or remote batch). If there is a heavy on-line user load and as the number of batch job classes increases, the number of compute-bound tasks will increase. Batch jobs will get more CPU time due to the large amount of time assigned to them. More attention can be given to certain categories of batch jobs by increasing the number of batch job classes suitable for them. This procedure makes no significant difference in on-line response time because interactive requests have a higher priority than compute-bound jobs.

SYSTEM TUNING AND MEASUREMENT

The CP-6 system includes a comprehensive set of performance measurement and system control facilities. These facilities allow the system manager to determine how the system is performing and to adjust critical operational parameters to achieve better performance. The CONTROL and STATS processors provide these facilities.

The CONTROL processor provides a means of adjusting system performance. CONTROL processor commands enable the system manager to display measurements and to 'tune' the system as needed by setting new values for parameters that affect system performance. CONTROL provides commands for:

- Display of system parameters.
- Modification of system control parameters.
- Display and modification of batch job class definitions.

The STATS processor performs two functions: displaying selected performance data in real-time, and creating 'snapshot' records of performance data for later processing. The STATS processor provides a global view of system performance by formatting and displaying the statistical data collected as snapshots. The processor allows the system manager to:

- Request a chronological listing of snapshot data for one or more groups of performance statistics.
- Specify a filter to remove out-of-range data from the sample for subsequent reports.

RESOURCE MANAGEMENT

The term resource has a very specific meaning in the following discussion. A resource is any portion of the CP-6 installation that is to be shared by the users in such a manner that each user requiring the resource is allocated the resource for its exclusive use. Private disk packs are an exception. Under some circumstances, private disk packs may be shared even though they have been defined to be resources. Tapes, disks, printers, the CPU(s), and main memory are common types of resources. Spooled devices and public storage devices can never be defined to be resources because they are non-allocatable devices; that is, they are never reserved for the exclusive use of one user.

Special resource management routines within the monitor keep track of the number of resources of each kind that are available for use. For a batch job, the requirement for resources is compared with the available resources and the job is not started unless sufficient resources are available. Furthermore, the resources are reserved for the exclusive use of the job. Thus availablity of resources is guaranteed even if time elapses between job startup and actual use of the resources.

The CP-6 system requires no correspondence between a physical device and a managed resource. When there is no correspondence between a resource and an actual physical device, the resource is called a pseudo-resource. Pseudo-resources are used to achieve special job scheduling effects and for other purposes.

The system manager must define the installation's resources, establish system defaults and maximums for use of the resources, and set limits on the use of the resources for the individual users.

During system definition, the system manager establishes which items are to be considered as resources. For each resource, the system manager establishes the system defaults and maximum values.

The CONTROL processor is used to dynamically modify the default and maximum values associated with each resource. Resources are defined during system definition, but a resource may be effectively removed from or returned to the system by appropriate modification of the values associated with the resource.

The SUPER processor is used to establish the maximum amount of each resource that is to be available to each particular user.

USER AUTHORIZATION

Before any user can perform any CP-6 processing, the user's account must be created by the system manager. When the account is created, the system manager must specify the user's name and account number. In addition to these items, the system manager decides for each account:

1. The associated type and level of privilege granted the user. The user may be authorized for use of many facilities. For example, the user may be authorized to:

   - Run diagnostic programs.

   - Access and change the monitor.

   - Read and write error files; request devices; invoke diagnostics, and authorize enqueue/dequeue automatically.

   - Examine (but not change) the monitor.

2. Whether an initial password is to be associated with the account. If specified, log-on cannot be completed unless the password is provided.

3. Whether all files created under this account may be read, executed, or modified by other users. A default is applied to files created by the user unless the user explicitly gives overriding instructions.

4. Whether a security check is to be performed on newly allocated main memory to be used by this account. If requested, all memory that the user will access will be effectively erased before being accessed.

5. Whether the processors available to this account are to be restricted.

6. Whether to automatically connect a user of this account to a given processor.

Through these features, an installation has numerous security controls over each and every user. These controls may, at the system manager's discretion, be applied to users on an individual account basis.

PROJECT ADMINISTRATION

To ease the task of user account management, the system manager may delegate authority to project administrators. The system manager defines the constraints and maximum resources that a project of users may have. A project administrator may then define each user within the project and, with appropriate authorization, may create sub-projects within the project.

USE ACCOUNTING

During the operation of each job, the CP-6 system accumulates a wide assortment of accounting information which collectively records the job's activities. Accounting statistics gathered include counts of CPU use, memory use, I/O operations, pages printed, cards punched, monitor service requests, terminal I/O character rates, and many others. These accounting statistics are written into a file which may be used by the installation to prepare charges for its customers. Interfaces are provided so that installation-supplied routines may augment or modify the records written to the accounting file. Furthermore, extra counters are included for use by the installation in preparing special charges of their own, either for unique programs or for individual transactions within unique programs.

An option exists which will cause the system to write an accounting record for each job step completed. This record includes the counter values attributable to the step plus the name of the executing program. Use of this option facilitates charging for proprietary program products.

The RATES processor allows the system manager to define monetary charges for each of the system counted values. Given these values, the system will automatically calculate the proper charges for the user session or job step. Additional features include a currency conversion multiplier, different rate structures to be applied to different classes of users and to the same users under different circumstances such as time of day, and charge discounting. Separate charging schedules are available for printed forms and for program usage. Program interfaces within the charge calculating program permit the addition of installation-specific routines into the charge calculation process.

The system provides summary information to batch and on-line users at the end of each job, detailing the counter values accumulated. Charges are included if the RATES processor is used, and details at each job step are optionally available.

CP-6 budget accounting is also available. Budget accounting permits an installation to establish a budget hierarchy and to control access to the system depending on a user's remaining budget. Budget accounting controls determine whether or not a user (or anyone else) will be denied access when the budget is exhausted. The installation has the option of performing budget calculation at step-time, providing very tight control on budget over-runs. A job-step which exhausts the budget may be, at installation option, the last one the user is allowed. Additional granule accounting is available to describe inactive files and shelf life as well as mounted life of pack sets. The budget accounting hierarchy follows the user project hierarchy in the user authorization file.

Section 18

CP-6 Computer Operations

The CP-6 system is designed to function effectively with a minimum of operator intervention. System support requirements are minimized during initial installation. Reconfiguration and patching are simply and inexpensively implemented. Since recovery is automatic, the system can be run in unattended mode without an operator present when specific applications do not demand the use of central site peripherals such as tape drives and line printers. Any local or remote terminal can be designated as an operator's console, thereby eliminating problems caused by console failure. Operator tasks can be divided into logical groups, each handled through a separate terminal, thus reducing the training required for an operations staff. Each of these features result in significant operations cost reduction.

INSTALLING, RECONFIGURING, AND PATCHING THE SYSTEM

CP-6 installation, reconfiguration, and patch features are designed to minimize the time and staff required to install or modify the system. The procedures have been standardized and simplified so that virtually no support is required from Honeywell staff.

Any hardware or software configuration can be installed completely by the customer's system support staff. After the hardware has been installed and tested, a support staff of one can install the full set of CP-6 software within two hours, including both central system and separately priced components.

The CP-6 operating system arrives at the site on a minimum number of system tapes (four at most) that contain all the ordered components of the system including separately priced ones. If the system configuration is standard, the support staff need only mount the tapes, push the boot buttons, and follow the instructions provided by Honeywell. If the system is configured in a non-standard way, the procedure is only slightly modified.

Using the CP-6 SPIDER and PCL processors, the system support staff can install new or replacement processors in minutes, normally without creating new system tapes and on a running system. Reconfiguration can be accomplished without interrupting user service.

Honeywell-supplied system patches are likewise easily installed through patch files (at sites with synchronous communication capabilities) or patch tapes. Patches to processors may be installed on a running system without interrupting user service.

UNATTENDED SYSTEM OPERATION

An important feature of the CP-6 system is that the computer operator may leave the system alone and let it run itself. This feature allows an installation to have selected periods of time (for example, graveyard shift) to provide time-sharing or to run time-sharing concurrently with batch jobs which require no peripheral device action on the part of the operator.

To allow unattended operation, the operator uses a system facility to logically remove the peripheral devices which would require an operator's attention (e.g., line printers and tape drives). These devices can then be turned off so there need be no concern about a tape or printer device failure in the operator's absence. Printer output will collect in the output spooling files. When the operator returns, the devices can be turned on and returned to the system and the collected output will be printed. If an on-line or batch job requests the mounting of a tape, the request will be denied and only that one job will be affected. The system continues operation in a normal mode.

INITIALIZATION

Several procedures combine to cover the general subject of system start-up, initialization, and recovery from various levels of error situations. Each of the procedures is tailored to restoration of the minimum amount of the system required to regain operation. Further, recoveries proceed automatically, generally requiring no operator intervention.

JOB AND SYSTEM CONTROLS

The operator controls system operation through the use of console keyins. CP-6 operators may use any remote or local terminal to control system activity. Operator activities may be separated into several groups with each group of commands and their associated messages handled from a separate terminal (e.g., one for tape mounts, one for printer control).

A terminal that has been logged on as a console can simultaneously be used as a time-sharing terminal via a special set of keyins provided by the console ghost facility. The operator still receives messages and action requests, and can respond with device and other keyins, while acting as a time-sharing user. This feature assures maximum utilization of terminal resources.

REMOVABLE STORAGE INITIALIZATION

The Pack Set Initializer (PIG) program initializes pack sets for use with the file management system. PIG is used to establish serial numbers, account directories, and granule allocation and to write headers and other system information on selected areas of the volumes.

PERIPHERAL DEVICE ERROR PROCEDURES

If the monitor encounters an abnormal condition during an I/O operation, it will send a message to the operator. These messages are generated both for errors that are irrecoverable and for errors that are recoverable with operator assistance. The operator may respond with a device keyin of the form:

```
action device
```

The 'action' can be any of the following:

Action	Meaning
CONTINUE	Continue, the problem has been solved.
ERROR	Continue, but inform the program of the error.
RETRY	Retry the I/O operation.

In addition to logging errors on the operator's console, the system also maintains a system error log file. This file contains a log of system and peripheral device failures that were corrected, that were irrecoverable, or that required operator assistance for recovery.

Appendix A

IBEX Command Summary

This appendix contains a table of the commands interpreted by the IBEX processor. The commands are listed alphabetically, and a brief description of command function is included for each command.

Table A-1. IBEX Commands

Command Mnemonic	Function
ACCEPT	Controls printing of operator originated messages at the user's terminal.
ACQUIRE	Requests additional resources.
ADJUST	Modifies DCB assignments during a job step.
ATITLE	Inserts a title into the accounting record generated for a job, or deletes a previously assigned title.
BACKUP	Qualifies a file to be saved on backup tape storage.
BATCH	Submits one or more jobs for batch execution.
BUILD	Invokes the EDIT processor to create a file.
BYE	Terminates an on-line session and disconnects the terminal. The OFF command is a synonym.
CANCEL	Deletes a job from the batch job queue, aborts the processing of an exeucting job, or deletes a job's output from the output queue.
CHECK	Shows the status of jobs in the batch queue, jobs executing and jobs awaiting output.
COMMENT	Controls the listing of diagnostic output.
CONTINUE	Resumes an interrupted activity, and terminates interrupt mode. The GO and PROCEED commands are synonyms.
COPY	Transfers data between peripherals.
DATE	Requests a display of the time and date. The TIME command is a synonym.
DEFAULT	Establishes or rescinds default data replacement specifications.
DELETE	Deletes disk files.
DELTA	Invokes DELTA after execution is interrupted.

Table A-1. IBEX Command (cont)

Command Mnemonic	Function
DIRECTORY	Changes the default account and pack set for fids specified during the session or job.
DISPLAY	Prints information on current users and the system.
DONT	Cancels the conditions set by the ACCEPT, COMMENT, ECHO, LIST, or PROTECT command.
E	Invokes the EDIT processor to manipulate a file.
ECHO	Prints IBEX commands contained in a command file as they are read from the command stream.
END	Terminates interrupt mode, prompting the user for an IBEX command. The QUIT and STOP commands are synonyms.
ERASE	Deletes output accumulated for logical devices.
GET	Recalls a saved activity from disk storage.
GLOBAL	Establishes, rescinds and displays global data replacement specifications.
GO	Resumes an interrupted activity and terminates interrupt mode. The CONTINUE and GO commands are synonyms.
GOTO	Directs branching forward within a command stream.
IF	Establishes conditions for affecting command stream logic, and specifies alterations in the command stream flow.
JOB	Defines a batch job and its attributes.
L	Prints a summary of disk or labeled tape storage.
LDEV	Defines logical devices and also defines and modifies logical device attributes.
LET	Sets a value for STEPCC or defines a command variable and sets a value for it.
LIMIT	Establishes maximum values for system resources.
LINK	Invokes the LINK processor to create a run unit. The LOAD and LYNX commands are synonymous.
LIST	Determines disposition of listing output.
LOAD|LYNX	Invokes the LINK processor to create a run unit. The LINK command is a synonym.
MAP	Invokes the LINK processor to produce a map.
MESSAGE	Directs a message to the installation operator's console.
MODIFY	Modifies the name, password, and attributes of one or more files. The RENAME command is a synonym.
OFF	Terminates an on-line session and disconnects the terminal. The BYE command is a synonym.

Table A-1. IBEX Commands (cont)

Command Mnemonic	Function
ORESOURCE	Requests resources and establishes global limits for an on-line session.
PASSWORD	Creates, changes, or deletes a log-on password for an account.
PLATEN	Defines the number of lines per page and characters per line on terminal output.
PMD	Invokes DELTA to dump specified portions of a program which terminate abnormally.
PRINT	Immediately sends output accumulated for logical devices to their destinations.
PRIORITY	Establishes the default priority for a job.
PRIV	Requests authorization and deauthorization of privileges.
PROCEED	Resumes an interrupted activity and terminates interrupt mode. The CONTINUE and GO commands are synonyms.
PROFILE	Selects a terminal profile.
PROTECT	Instructs IBEX to issue a QUIT? message prior to performing any activity which will make the user's interrupted process unresumable.
QUIT	Terminates interrupt mode, prompting the user for an IBEX command. The END and STOP commands are synonyms.
RELEASE	Deallocates allocated resources.
REMOVE	Requests that a tape be removed from a system tape drive.
RENAME	Modifies the name, password, and attributes of one or more files. The MODIFY command is a synonym.
REPORT	Specifies the level of accounting statistics to be displayed after each job step.
RESET	Modifies DCB assignments at a job step.
RESOURCE	Requests resources and establishes global limits for batch processing.
REWIND	Rewinds a tape to its beginning or load point.
RUM	Invokes DELTA to process permanent patches against a run unit.
RUN	Invokes the LINK processor to link specified object units into a temporary run unit, fetches the run unit, and initiates execution.
rununit	Fetches the specified run unit and initiates execution.
SAVE	Copies the current contents of memory to a disk file.
SET	Assigns a file or device to a DCB and sets DCB parameters.

Table A-1. IBEX Commands (cont)

Command Mnemonic	Function
SETUP	Specifies an IBEX command that will be automatically executed whenever the issuing account logs on to the system.
START	Fetches a run unit and either initiates execution of it or invokes a debugger to process it.
STATUS	Displays information about system usage during the current session.
STOP	Terminates interrupt mode, prompting the user for an IBEX command. The END and QUIT commands are synonyms.
SWITCH	Sets and resets sense switches.
TABS	Sets terminal tab stops.
TERMINAL	Defines the attributes for a terminal.
TIME	Requests a display of the current time and date. The DATE command is a synonym.
TITLE	Inserts a heading at the beginning of each output listing page.
UNDER	Invokes a debug processor.
XEQ	Initiates execution of a file of commands.

Appendix B

DELTA Directive Summary

This appendix consists of the following six tables that describe the directives interpreted by the DELTA debug procssor:

- Table B-1 contains a list of DELTA housekeeping directives, which influence the behavior of the DELTA processor. These directives control I/O, addressing and symbols, stored directives management, and faults and traps.
- Table B-2 contains a list of DELTA execution control directives, which determine when DELTA is to assume control of an execution run unit. These directives control procedure and data breakpoints, transfers, procedure stepping, and special purpose execution.
- Table B-3 contains a list of DELTA execution tracing directives, which cause the flow of control within a run unit to be recorded and displayed.
- Table B-4 contains a list of DELTA memory display and modification directives, which display and change the control of both memory and program visible registers.
- Table B-5 contains a list of DELTA mode control directives, which instruct DELTA to change from the normal debug mode to:
  - o RUM mode to apply permanent patches to a run unit.
  - o ANLZ mode to examine the running monitor or a system dump file.
- Table B-6 contains a list of DELTA miscellaneous directives (those that do not fall into the other DELTA directives categories).

In each table, directives are listed alphabetically, and a brief description of directive function is included for each directive.

Table B-1. Housekeeping Directives

Directive	Function
ACTIVE/INACTIVE	Activates or deactivates a single directive or a range of stored directives.
ALTERNATE VARIABLES	Specifies alternate debug schema to be searched when an unqualified variable reference is not satisified by searching the current schema.
BYPASS	Bypasses assembler program units during stepping.
COPY	Causes DELTA output to be copied on the user terminal when the specified destination for output is other than the user terminal.

Table B-1. Housekeeping Directives (cont)

Directive	Function
DEFINE	Associates a value or location with a symbol.
DO	Executes the attachments to a stored directive or group of directives identified by the SAVE directive.
ECHO	Causes input to be echoed to an output device when DELTA input is from a device other than an on-line terminal.
EOM	Sets or resets a special activation (end of message) character set.
FORMAT	Specifies default format for MODIFY and EVALUATE display output.
KEEP/TRAP/IGNORE	Directs DELTA's handling of asynchronous events and other exceptional conditons.
KILL	Deactivates a toggle or removes a stored directive or a range of stored directives.
ON ABORT	Specifies activities to occur upon abort.
ON EXIT	Specifies activities to occur upon normal exit.
OUTPUT	Specifies destination for DELTA output.
PROMPT	Sets the DELTA prompt character.
RANGE	Specifies a range of offsets from a defined symbol to be used for position reporting.
READ	Causes DELTA to read other than the normal input stream.
REPORT	Directs DELTA's formatting of position reporting.
SAVE	Stores and remembers a single or a range of stored directives.
SCHEMA	Activates or deactivates schema usage or sets "current" schema.
SHOW	Displays the status of toggled options, keyword option or a single directive/attachment or range of stored directives and attachments.
SILENT/UNSILENT	Activates or deactivates the reporting of a single directive or a range of stored directives.
SYNTAX	Allows explicit specification of input syntax (for example, FORTRAN, COBOL, and RPG II).
USE NODE	Activates schema(s) associated with a specific overlay node. In RUM mode, allows access to specific overlay nodes.

Table B-2. Execution Control Directive

Directive	Function
ALIB	Specifies return/altreturn from M$ALIB call to DELTA.
AT	Sets an instruction breakpoint.
BREAK	Passes control to user interrupt routine.
EXIT	Exits from a run unit invoked by M$LINK and returns to the linking program, or continues an M$LDTRC or M$SAVE.
GO	Proceeds with program execution.
GOSTEP	Goes to a specified location and executes one step.
GOTRAP	Passes control to user's event handling routine when DELTA has been entered for an exceptional or asynchronous event.
GOTRAPSTEP	Passes control to user's event handling routine for execution of a single step.
ON CALL	Sets breakpoints on a specific procedure call.
ON CALLS	Sets breakpoints on all procedure calls.
ON NODE	Sets a breakpoint on a specific overlay.
ON NODES	Sets breakpoints on all overlays.
SOC	Steps one CALL statement, halting upon return.
STEP	Steps by statement or instruction.
WHEN	Sets a data breakpoint.
XCON	Passes control to the unit's exit control procedure simulating an exit condition.

Table B-3. Execution Tracing Directives

Directive	Function
HISTORY	Displays contents of the history buffer (filled by TRACE).
PLUGH	Traces back through the automatic stack and lists the return addresses leading to the arrival at the current procedure point.
TRACE	Traces transfers at the statement, substatement, or instruction level or traces the flow of paragraphs.
TRACE XCALLS	Traces entry to all procedures. If XCALLS is specified, traces entry to external procedures only.

Table B-4. Memory Display and Modification Directives

Directive	Function
DISPLAY	Displays the value of a variable or the contents of an address.
DUMP	Dumps a specified range of memory in octal or hexadecimal format. Optionally provides ASCII translation.
EVALUATE	Evaluates an expression and reports its value in a specified format. Reports the address of a program entity by segment and offset.
FIND	Searches memory under mask and optionally substitutes under mask.
LET	Changes the value of a variable or the contents of an address.
MODIFY	Displays the contents of an address and optionally replaces it with new contents.
PMD	Dumps specified portions of a program which terminates abnormally.
STORE	Modifies a range of memory. Optionally performs the modification under mask.

Table B-5. Mode Control Directives

Directive	Function
ANLZ	Associates the schemas for the CP-6 monitor and sets DELTA's domain of reference to that of the running monitor or a specified system dump file.
RUM	Invokes the Run Unit Modification (RUM) mode for permanently patching a run unit file.

Table B-6. Miscellaneous Directives

Directive	Function
END|QUIT	Unconditionally exits to the command processor.
HELP	Provides HELP for most commands.
LIST	Lists changes made during Run Unit Modification.
PROTECT	Sets Protect mode (disallows LET, MODIFY store).
SAD	Allows addressing through the monitor Special Access Descriptor for privileged users.
UNFID	Performs M$UNFID on specified DCB.
XEQ	Executes a GMAP6 assembler instruction.
?	Requests elaboration of last-issued error or HELP message.
??	Requests all available information on the last-issued error or HELP message.

Appendix C

EDIT Command Summary

This appendix consists of the following three tables that describe the commands interpreted by the EDIT processor:

- Table C-1 contains a list of EDIT file commands, which build, copy, merge and delete files. EDIT file commands are organized functionally into the following groupings: editing attribute commands, file manipulation commands, and informational commands.
- Table C-2 contains a list of EDIT record commands, which insert delete, reorder, replace and print the lines of a file. EDIT record commands are organized functionally into the following groupings: alter commands; insert/delete commands; print commands; reorder commands; search commands; and select commands.
- Table C-3 contains a list of EDIT intra-record commands, which modify the characters within a record. EDIT intra-record commands are organized functionally into the following groupings: conditional execution commands, string manipulation commands, and miscellaneous commands.

In each table, commands are listed alphabetically within functional groupings. A brief description of command function is included for each command.

Table C-1. EDIT File Commands

Functional Grouping	Command	Description
Editing Attribute Commands	BP	Retain fields of blanks.
	CR	Include carriage returns.
	CRPT	Set a data encryption seed.
	RP	Maintain record lengths.
	TA	Set tab positions.
	TABC	Controls tab compression.
	TABX	Controls tab expansion.
	TYPE	Select a file type code.
	VE	Verify editing.
File Manipulation Commands	BUILD	Create a new file.
	COPY	Make a copy of a file.
	DELETE	Delete a file.
	EDIT	Select a file for editing.
	END	Return to IBEX.
	EXAMINE	Select a file for examination.
	MERGE	Merge one file into another.
	READ	Read a command file.
Informational Commands	HELP	Supply information.
	LIST	List files.
	STATUS	Display EDIT status.

Table C-2. EDIT Record Commands

Functional Grouping	Command	Description
Alter Record Commands	AD	Add to end of record.
	CM	Insert commentary.
	CT	Print and insert commentary.
	RR	Reread record.
Insert/Delete Record Commands	AP	Append records.
	DE	Delete records.
	IA	Insert after records.
	IB	Insert before records.
	IN	Insert records.
	IP	Insert new records.
	IS	Insert records with period prompt.
Print Record Commands	OL	Print at the line-printer.
	TC	Print records compressed.
	TN	Print next record.
	TP	Print previous record.
	TS	Print records without keys.
	TY	Print record with keys.
Reorder Record Commands	MD	Move and delete records.
	MK	Move and keep records.
	RN	Renumber records.
Search Record Commands	FD	Find and delete records.
	FS	Find and print keys of records.
	FT	Find and print records.
Select Record Commands	SE	Select records for intra-record editing.
	SS	Select records for set and step.
	ST	Select records for set, step, and type.

Table C-3. EDIT Intra-Record Commands

Functional Grouping	Command	Description
Conditional Execution Commands	EI	End condition.
	EL	Else conditional execution.
	IF	Conditional execution.
	QP	Quit processing, return.
	RL	Return to beginning of command line.
String Manipulation Commands	A	Align columns.
	D	Delete string.
	E	Overwrite string, blank fill.
	F	Insert string following.
	L	Shift image left.
	O	Overwrite string.
	P	Insert string preceding.
	R	Shift image right.
	S	Substitute string.
Miscellaneous Commands	CI	Copy current record, interlacing.
	CL	Override column editing limits.
	CP	Copy current record, protected.
	JU	Jump to new sequence.
	NO	No change.
	RF	Reverse blank preservation.
	TX	Print changed records.

Appendix D

PCL Command Summary

This appendix contains a table of the commands interpreted by the PCL processor. The commands are listed alphabetically, and a brief description of command function is included for each command.

Table D-1. PCL Commands

Command	Function
COPY	Transfers data between peripherals.
COPYALL	Performs mass file transfers from one account to another.
COPYSTD	Performs mass file transfers where the sources are specified in an STD file.
DELETE	Deletes disk files.
DELETESTD	Deletes disk files as defined in an STD file.
END	Terminates PCL processing and returns the user to IBEX.
ERASE	Deletes device streams from the system.
ERRORS	Determines the handling of output files and streams in the event of an error or break condition.
HELP	Provides information about PCL, its concepts, and commands.
LIST	Prints a summary of disk or labeled tape storage.
LISTSTD	Prints a summary of disk or labeled tape STD file storage.
MODIFY	Modifies the name, password and/or file attributes of one or more files.
PRINT	Prints all pending output through device streams.
RELEASE	Requests that a tape be removed from the system tape drives and that the drive be released to the system.
REMOVE	Requests that a tape be removed from a system tape drive.
REVIEW	Prints a summary of disk storage, allowing the user to user to delete, relist or copy the contents of the file.

Table D-1. PCL Commands (cont)

Command	Function
REVIEWSTD	Prints a summary of disk storage, with the files defined in an STD file, allowing the user to delete, backup, or copy the contents of the file.
REWIND	Rewinds a tape to its beginning or load point.
SCAN	Searches a free tape up to the first encountered double tape mark and reports on the density, number of records, and longest record length.
SPE	Positions a tape just beyond the end of its last file on labeled tape or between the next two adjacent tape marks on free tape.
SPF	Positions a tape to the beginning of a specified file.
SPR	Positions a free tape forwards or backwards a specified number of records.
TX	Sets the tab stops for PCL copy operations which use the TX option.
WEOF	Writes an end-of-file record to the specified device.

Appendix E

Summary of Monitor Services

This appendix contains a table of CP-6 monitor services. These services are available to user programs regardless of the language in which they are written; interface routines may be required in certain cases. The monitor services are listed alphabetically by name, and a brief description of service function is included for each monitor service.

Table E-1. CP-6 Monitor Services

Name	Function
M$ACPL	Accept coupling.
M$ACTIVATE	Allow other users and terminals access to a comgroup.
M$ALIB	Associate a monitor service or alternate shared library (or a debugger) with the user program.
M$ASUSER	Attach a suspended user image.
M$BADPP	Remove a bad page from normal use.
M$CGCTL	Establish comgroup parameters.
M$CGINFO	Get information about comgroup status.
M$CHECK	Check the I/O completion type.
M$CHGUNIT	Increment counters in the user JIT.
M$CLOSE	Close a file (terminate I/O through a DCB).
M$CLRSTK	Clear the Exceptional Condition Stack and proceed in line.
M$CORRES	Check for correspondence of DCB assignments.
M$COUPLE	Associate one terminal with another.
M$CPEXIT	Exit from a Command Processor.
M$CVM	Change the virtual map.
M$CVOL	Terminate I/O to the magnetic tape reel.
M$DCB	Compile a data control block (DCB).
M$DEACTIVATE	Disallow other users and terminals access to a comgroup.
M$DECOUPLE	End association of terminals.
M$DELREC	Delete data records or ranges of records.

Table E-1. CP-6 Monitor Services (cont)

Name	Function
M$DEQ	Dequeue for a logical resource.
M$DEVICE	Change device formatting attributes.
M$DISPLAY	Display system load parameters.
M$DISPRES	Display a list of resources currently owned by a program.
M$DLIB	Disassociate a core or alternate shared library, or a debugger from the user program.
M$DRTN	Return to the user from a debugger.
M$DSUSER	Delete a suspended user image.
M$ENQ	Enqueue for a logical resource.
M$EOM	Set the activation character set and read timeout.
M$ERR	Error the current job step.
M$ERRMSG	Send a message from an error message file.
M$EVENT	Give the user control at event completion.
M$EXIT	Exit to monitor normally.
M$EXTEND	Increase the file size.
M$FDP	Free dynamic pages.
M$FDS	Free a user data segment.
M$FEBOOT	Reboot Level 6 FEPs.
M$FECTL	Control front-end processor operation.
M$FEDUMP	Dump Level 6 FEP memory.
M$FEPDATA	Log FEP performance data.
M$FID	Set up a fid.
M$FSUSER	Check for a suspended program for a newly logged on user.
M$FVP	Free a virtual page.
M$GBPL	Get a bad page list.
M$GDDL	Get the dynamic data limits.
M$GDP	Get dynamic pages.
M$GDS	Get or enlarge user data segments.
M$GETDCB	Create a data control block (DCB) at run time.
M$GETMOUSE	Get the data segment of the PMME monitoring routine.

Table E-1. CP-6 Monitor Services (cont)

Name	Function
M$GETPM	Get general system performance monitoring data.
M$GETSTATE	Get the data segment of the state monitoring routine.
M$GJOB	Start the system ghost.
M$GLINEATTR	Get line (physical connection) attributes.
M$GOODPP	Return a page to normal use.
M$GTRMATTR	Get terminal attributes.
M$GTRMCTL	Get terminal control flags.
M$GTRMTAB	Get device (physical) tabs.
M$GVP	Get a virtual page.
M$HELP	Send a message from a HELP file.
M$INT	Give the user control at BRK keyin.
M$INTRTN	Return to user from alternate shared library when break control occurs.
M$IOQ	Perform a direct I/O request.
M$JOBSTATS	Control job status operations.
M$KEYIN	Cause a write, read, or write-followed-by-read (keyin) to the operator's console.
M$LDEV	Change the attributes of logical devices.
M$LDTRC	Transfer to a separate program with no possible return.
M$LIMIT	Reserve resources.
M$LINES	Get the number of lines remaining on the printer page.
M$LINK	Call a separate program with return expected.
M$MADMUCK	Associate an account with the pack set on which it resides.
M$MBS	Obtain resources for a batch job or determine what resources are available.
M$MERC	Give the monitor control to process monitor service error.
M$MERCS	A variation of M$MERC using the Exceptional Condition Stack.
M$MONINFO	Return information about the site and the running monitor.
M$MPL	Mark bad pages as in test mode.
M$OCMSG	Write console messages.
M$OLAY	Load or release a program overlay.

Table E-1. CP-6 Monitor Services (cont)

Name	Function
M$OPEN	Open a file (initialize DCB).
M$PFIL	Position to the beginning or end of the current file.
M$PLATEN	Set paper (forms) characteristics.
M$PRECORD	Reposition a disk or tape file.
M$PROCNAME	Return the name(s) of shared processor(s) associated with the program in execution.
M$PROFILE	Change the terminal profile.
M$PROMPT	Set the prompt string.
M$RCPL	Reject coupling.
M$RDSYSLOG	Read the system error log.
M$READ	Read a data record into a user buffer.
M$RELDCB	Release the space occupied by any closed DCB.
M$RELRES	Release resources owned by a program.
M$REM	Enable dismounting of a tape volume and, optionally, releasing of the resource.
M$RENV	Restore an M$SENV-saved environment.
M$REQUIRE	Ensure ownership of needed pseudo resources.
M$RETRY	Retry a monitor service request.
M$RETRYS	A variation of M$RETRY using the Exceptional Condition Stack.
M$REW	Rewind a tape or reposition a disk.
M$RPRIV	Reset privilege bits.
M$RSPP	Return a stolen physical memory page.
M$RSWITCH	Reset pseudo switches in the user JIT.
M$SAD	Set special access descriptors.
M$SAVE	Save a program memory image.
M$SCON	Set parameters for a SAVEd program.
M$SCREECH	Enter recovery.
M$SENV	Save the environment after a monitor service request error.
M$SETFMA	Change the file management default account and pack set name.
M$SETFP	Send a Forms Program to the front-end processor.

Table E-1. CP-6 Monitor Services (cont)

Name	Function
M$SINPUT	Set the effective last line typed by user.
M$SMOUSE	Initiate the PMME mounting feature.
M$SMPRT	Set memory protect.
M$SPRIV	Set privilege bits.
M$SSC	Set software control flags.
M$SSTATE	Initiate the user-state monitoring feature.
M$SSWITCH	Set pseudo switches in the user JIT.
M$STIMER	Set the timer interval and time-out entry.
M$STLPP	Get stolen physical memory pages.
M$STRAP	Simulate a trap.
M$STRMATTR	Set terminal attributes.
M$STRMCTL	Set terminal control flags.
M$STRMTAB	Set device (physical) tabs.
M$SYSCON	Partition, return and display status of devices.
M$TDCLOSE	Perform a test and diagnostic close.
M$TDIO	Perform a test and diagnostic input or output.
M$TDOPEN	Perform a test and diagnostic open.
M$TDREQCPU	Request CPU for test and diagnostics.
M$TIME	Get the current time.
M$TRAP	Give the user control on program traps.
M$TRMISC	Allow or inhibit operator sending and broadcasting to the terminal.
M$TRTN	Return normally to the user.
M$TRUNC	Release POOL buffers back to the system after I/O completion.
M$TTIMER	Get the current timer value and optionally cancel the current interval.
M$UMPL	Unmark bad pages.
M$UNFID	Convert components from a DCB into a fid.
M$UNLATCH	Release a latched transaction.
M$USRFIELD	Set JIT user fid.
M$WAIT	Wait a specified period of real time.

Table E-1. CP-6 Monitor Services (cont)

Name	Function
M$WEOF	Write an end of file (on free tape), an EOD (output to the card punch) or a top-of-form (output to a line printer).
M$WRITE	Write a data record from a user buffer to a file or device.
M$WRSYSLOG	Write an entry to the system error log.
M$XCON	Give the user control at program exit.
M$XCONRTN	Defer exit control processing by a special shared processor until all user exit control processing is completed.
M$XEQTIME	Get execution and service time expended for the current job.
M$XMOUSE	Terminate the PMME monitoring feature.
M$XSTATE	Terminate the user-state monitoring feature.
M$XXX	Abort the current job step.
M$YC	Simulate a CONTROL-Y sequence, giving control to the command processor.

Glossary

account - a group of files and privileges, usually associated with a particular user.

alternate shared library - a special shared processor that resides in its own working space, has greater privilege than the user program, and can be called directly from the user program (e.g., I-D-S/II).

ANS tape - a tape that has labels written in American National Standard (ANS) format.

application processor - a Honeywell supported processor intended for use in specific types of applications such as data base management.

bandwidth - the maximum rate at which memory, an I/O channel, or a front-end processor can deliver or accept information.

batch job - a job submitted to the batch job queue through the central site card reader, through an on-line terminal (using the BATCH command), or through a remote batch terminal.

batch job class - a logical entity used by the scheduler to select jobs from the batch job queue for execution. A job class is each type of resource that a job requires to be scheduled to runin that class. (Resources include such items as main memory, CPU availability tape drives, disk spindles, or pseudo resources.)

batch job queue - a set of jobs to be run in the batch mode. These jobs are scheduled by CP-6 in a manner that optimizes the use of nonsharable resources.

bipoint line - a line that connects a single remote transaction processing station to the computer center. (See multipoint line.)

block - a block of disk sectors large enough to contain 1,024 words (a page) of stored information.

block stamp - a one word item at the beginning of each granule in a file. It contains an identification of the file plus the low order bits of the granule number. The main function of the granule stamp is to facilitate system reliability by identifying the file to which each granule belongs.

comgroup - a CP-6 logical communications network commonly used to connect terminals to programs and programs to programs. Through this mechanism, a terminal may be accessed by name.

command processor - a processor which enables the users to direct the monitor to perform functions required for the processing of their jobs.

data control block (DCB) - a table in the user's program that contains the information used by the monitor in the performance of an I/O operation.

data set - a device which converts data processing device signals to telephone tones and telephone tones to device signals (also referred to as 'modem').

data set controller - a hardware interface between a remote processing terminal and the central computer.

DCB - see data control block.

execution control commands - commands that control job step construction and execution and provide communication between a program and its environment.

execution control language - commands that control program construction and execution of programs and provide communication between a program and its environment.

execution control processor - a processor used to load, run, and/or debug a user program.

FEP - front-end processor.

fid - the identification information of a CP-6 file.

file - a collection of data in one or more formats.

file information table - a table of information associated with each file. It contains such information as file type, size, location, access controls, and dates.

FIT - see file information table.

FPT - see function parameter table.

front-end processor - a minicomputer on to which the processing and communications load is distributed.

function parameter table (FPT) - a table through which a user's program communicates with a monitor function (such as an I/O function).

ghost job - a job which is neither a batch nor an on-line program. It is initiated and logged on by the monitor, the operator, or another job. It may consist of a single job step or it may be controlled by a file or the execution control language.

HASP - a communication protocol commonly used between central site computer centers and remote batch terminals. It includes 'multileaving', which provides the ability to combine data records for several destination devices into a single transmission block.

HDLC - a full-duplex, bit-oriented data transmission protocol. The Honeywell and IBM versions and the international standard are nearly identical.

JIT - see job information table.

job - a collection of steps or activities presented together to a data processing system for execution.

job information table - a table associated with each active job. The table contains accounting, memory mapping, and temporary monitor information.

job step - a subunit of job processing such as compilation, assembly, loading, or execution.

key - a data item that uniquely identifies a record within a keyed or indexed file.

keyin - information entered by the operator via a keyboard.

language processor - either a processor which translates assembly level or high-level source code into machine object code for execution or an interpreter of source code or commands.

library - a collection of frequently used routines in a form that facilitates their inclusion into programs.

linker - a processor that combines and transforms the output of one or more compilations into a single run unit.

logical device - a peripheral device represented in a program by a special name (e.g., SI or LO) rather than by specific physical device name.

logical device stream - an information stream which may be used when performing input from or output to a spooling device. Several logical device streams may be defined at system definition; each is given a name (e.g., LP01, CP01, CR01), each is assigned to a default physical device, and each is given default attributes. The user may perform I/O through a logical device stream with the default physical device and attributes, or he may change one or both to satisfy the requirements of his job.

modem - see data set.

monitor - a control program that supervises the processing, loading, and execution of other programs.

monitor services - operations performed by the monitor on behalf of a user program. (Also referred to as system services.)

multipoint line - a line that connects two or more remote terminal stations to the central computer. A line controlled by the computer as though it were connected to two or more stations is considered to be multipoint even though it connects only one station to the computer. (See bipoint line.)

object file - a file consisting of one or more object units. Object files serve as input to the linking processor.

object unit - the series of records containing the instructions, debug schema, and linking information pertaining to a single program or subprogram (i.e., from the beginning to the end). An object unit is the output of a compilation or an assembly.

overlay program - a tree-structured program in which the node currently being executed may overlay the storage area occupied by a previously executed node.

packset - a group of disk packs associated with a unique identifier.

processor - a public program supplied by Honeywell. See application processor, command processor, language processor, shared processor, special shared processor, shared run-time library, standard shared processor, system management processor, and utility processor.

prompt character - a character sent to the terminal by an on-line program to indicate the next line of input may be entered.

protected mode - a mode of tape protection in which only expired ANS tapes of the specified label may be written; all ANS tapes must be initialized by the LABEL processor; no tape serial number specification is allowed at the operator's console; specification of an output serial number forces processing to be done only on a tape already having that serial number. (See 'semi-protected mode', 'unprotected mode'.)

public library - a set of library routines declared to be public (i.e., to be used in common by all concurrent users).

recovery - restart of the system after a temporary halt in system performance.

resource - a portion of a CP-6 installation to be shared by users in such a manner that each user requiring it has it allocated for his exclusive use.

run unit - a memory image of an executable program. It is the result of the linking process and is executed as a job step.

SDLC - a full-duplex bit-oriented data transmission protocol. The Honeywell and IBM versions and the international standard are nearly identical.

secondary storage - any rapid-access storage medium other than main memory (e.g., disk storage).

semi-protected mode - a mode of tape protection in which a warning is posted to the operator when output is attempted on an unexpired ANS tape. The operator can authorize the overwriting of the tape through a keyin. ANS tapes may be initialized by the LABEL processor or may be given labels as the result of an operator key-in; tape serial number specification is allowed at the operator's console; and specification of an output serial number forces processing to be done only on a tape already having that serial number unless the operator authorizes an overwrite. (See 'protected mode', 'unprotected mode'.)

shared processor - a processor which permits the sharing of the code among all simultaneous users. Each user of a shared processor has its own copy only of the data and DCB portion of that program; the procedure (code) portion is shared by all users associated with the shared program.

shared run-time library - a shared processor mapped into the user's working space quarter along with the user program (e.g., FORTRAN Run-Time Package).

software check - a failure that could have an adverse effect on the system or its programs. It causes the system to go into an automatic recovery procedure.

source language - a language used to prepare a source program suitable for processing by an assembler or a compiler.

special name - a symbolic name used to identify a logical system device.

special shared processor - a shared processor that resides in its own working space but can be called to execute in conjunction with the user program (e.g., DELTA).

spooling - the technique of buffering unit record input or output on disk to allow simulated unit record I/O. User programs thus proceed at speeds unlimited by the speed of unit record peripherals.

standard shared processor - a shared processor mapped into the user's working space and effectively is part of the user program (e.g., FORTRAN compiler).

symbiont - see spooling.

system definition - the process of creating an operating system tailored to the specific requirements of an installation. The major steps in a system definition include: gathering the relevant programs, generating specific monitor tables, loading monitor and system processors, and writing a bootable system tape.

system management processor - a processor that performs some function that provides the manager of a CP-6 installation with on-line control of the system.

system services - operations performed by the monitor at the request of a user program. (Also referred to as monitor services.)

unit record equipment - peripherals which deal with hard copy media such as card readers, card punches, and printers.

unprotected mode - a mode of tape use in which both unexpired and expired labeled tape can be overwritten without operator intervention. (See 'protected mode', 'semi-protected mode'.)

utility processor - a processor that performs some general function required by users for running and using the CP-6 system. Examples of service processors are EDIT (which enables the user to build and manipulate files of program and data) and PCL (which enables the user to move files among card devices, line printers, disk, etc.).

working space - the megaword of main memory available to every user. Other working spaces are used by the system in carrying out user associated services.

Index

HONEYWELL INFORMATION SYSTEMS
Technical Publications Remarks Form

TITLE	CP-6 CONCEPTS & FACILITIES	ORDER NO.	CE26-01
		DATED	SEPTEMBER 1980

ERRORS IN PUBLICATION

SUGGESTIONS FOR IMPROVEMENT TO PUBLICATION

Your comments will be investigated by appropriate technical personnel and action will be taken as required. Receipt of all forms will be acknowledged; however, if you require a detailed reply, check here. ☐

FROM: NAME ______________________ DATE ____________

TITLE ______________________

COMPANY ______________________

ADDRESS ______________________

____

PLEASE FOLD AND TAPE–
NOTE: U. S. Postal Service will not deliver stapled forms

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 39531 WALTHAM, MA02154

POSTAGE WILL BE PAID BY ADDRESSEE

HONEYWELL INFORMATION SYSTEMS
200 SMITH STREET
WALTHAM, MA 02154

ATTN: PUBLICATIONS, MS486

Honeywell

Honeywell

Honeywell Information Systems
In the U.S.A.: 200 Smith Street, MS 486, Waltham, Massachusetts 02154
In Canada: 2025 Sheppard Avenue East, Willowdale, Ontario M2J 1W5
In the U.K.: Great West Road, Brentford, Middlesex TW8 9DH
In Australia: 124 Walker Street, North Sydney, N.S.W. 2060
In Mexico: Avenida Nuevo Leon 250, Mexico 11, D.F.

29298, 21080, Printed in U.S.A.

CE26-01