CONTROL PROGRAM-SIX (CP-6)

CONCEPTS AND FACILITIES

AUGUST 1977

LADC # 874

PRELIMINARY DRAFT

PREFACE

Honeywell Control Program-Six (CP-6) is a migration product designed to provide users of Xerox Control Program-Five (CP-V) with a highly compatible growth path to Honeywell state-of-the-art hardware. Therefore this manual contains comparative references to CP-V for the benefit of users familiar with CP-V.

GLOSSARY

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

batch stream   a logical entity which is used by the scheduler to select jobs from the batch job queue for execution. A batch stream is essentially a definition of the minimum and maximum number of each type of resource that a job must require to be scheduled to run in that batch stream. (Resources include such items as main memory, CPU time, tape drives, disk spindles, and other peripheral devices.) Typically 32 batch streams are defined by the system manager. This allows 32 batch jobs to be run concurrently. When a batch stream does not have a job scheduled for it, the scheduler selects a job from the batch job queue which fits the resource profile of that batch stream and schedules the job for execution in it.

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

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   1) a device which converts data processing device signals to telephone tones and telephone tones to device signals. (Also referred to as "modem".); or 2) an organization of data or files.

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

DCB   see data control block.

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

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.

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 that is neither a batch nor an on-line program. It is initiated and logged on by the monitor, the operator, or another job and consists of a single job step. When the ghost program exits, the ghost terminates.

granule   a block of disk sectors large enough to contain 1024 words (a page) of stored information.

granule stamp   a one word item at the beginning of each granule in a file. It contains an id which is the same for each granule in a file and the low order bits of the granule number. The main function of the granule stamp is to facilitate system reliability.

HASP   a communications protocol commonly used between central site computer centers and remote batch terminals. It includes "multileaving" which the ability to combine data records for several destination devices (printers, punches, etc.) in a single transmission block.

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

JCL   the job control language consisting of commands of the Interactive and Batch Executive (IBEX) processor.

JIT   see job information table.

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

job control commands (JCL)   commands that control the construction and execution of programs and provide communication between a program and its environment.

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

job step   an executable computer program and 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.

key-in   information entered by the operator via a keyboard.

language processor   either a $\overset{\text{Program}}{\wedge}$ 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 expedites their inclusion into other 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 that is represented in a program by an operational label (e.g., SI or LO) rather than by specific physical device name.

logical device stream   an information stream that may be used when performing input from or output to a symbiont device. Several logical device streams may be defined at System Definition. Each logical device stream is given a name (e.g., L1, P1, C1), 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 the physical device and/or attributes to satisfy the requirements of his job.

MAILBOX file   a file with the name MAILBOX which contains account-specific messages from the system. Each account may have a MAILBOX file associated with it.

modem   see data set.

monitor   a 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 transaction processing 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.

op label   see operational label.

operational label   a symbolic name used to identify a logical system device.

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

processor   a public program supplied by Honeywell.

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

protective mode   a mode of tape protection in which only ANS expired tapes 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; tapes mounted as IN may not be written; and tapes mounted as other than IN must have a write ring. (See "semiprotective mode".)

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

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

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

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

semi-protective mode a mode of tape protection in which a warning is posted to the operator when output is attempted on an unexpired ANS tape, or when a tape mounted as OUTPUT has no write ring. The operator can authorize the overwriting of the tape or the override of OUTPUT through a key-in. 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 "protective 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 that is mapped into the user's working space quarter along with the user program (e.g., FORTRAN Run-Time Package).

software check a software failure that could have adverse effect on the system. 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 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 record peripherals.

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

stream-id the name of a logical device stream.

symbiont a monitor routine that transfers information between disk storage and a peripheral device independent of and concurrent with job processing. (See spooling.)

system definition the process of creating an operating system that is 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 system manager of a CP-6 installation with on-line control of the system.

system services   operations performed by the monitor on behalf 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.

utility processor   a program processor that performs some general function required 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.).

TABLE OF CONTENTS

SECTION I

INTRODUCTION

CONTROL PROGRAM-SIX

Control Program-Six (CP-6) is a comprehensive, multi-use virtual memory operating system that provides the following six modes of operation:

- Time-sharing
- Multiprogrammed batch processing
- Remote batch processing
- Network processing
- Distributed real-time processing
- Transaction processing

CP-6 supports these modes of operation with balanced service. There is no inherent emphasis on any one mode of processing.

The six modes of operation are also designed so that they may operate concurrently. That is, several programs utilizing different modes may be simultaneously resident in main memory with CP-6 selecting the appropriate one for execution at a given time or possibly selecting more than one if the hardware configuration provides more than one central processing unit.

Modularity allows the user to select only the mode or modes required by a given task. CP-6 performs equally well whether a single mode is used or multiple modes are combined. Common services, file management and processors facilitate an exceptional degree of compatibility between all the modes of operation. These common services and the fact that they are provided in a rather uniform way to all users regardless of the mode of use has been emphasized in the design of CP-6.

Before discussing the distinguishing characteristics of each mode, it is important to emphasize that the CP-6 functional elements are essentially the same for all programs regardless of mode of operation. The elemental building blocks of CP-6 are:

- Highly efficient scheduler
- File management system
- I/O and communication systems
- Command language

- Reentrant languages
- Service facilities

A comprehensive file system is integrated into CP-6. This means that programs can communicate easily since files are managed in a common way by a common set of programs. Also of great importance is the complete device independence provided by the logical I/O services so that files, devices, or terminals are all read and written with the same set of monitor service calls.

Modes of Operation

To give the reader a general overview of the system, the six modes of operation will be described briefly in this introduction.

- The time-sharing mode allows up to 500 interactive terminals to be connected to the central computer system at one time. Rapid access to and response from CP-6 creates an atmosphere in which each time-sharing user appears to have the entire system dedicated to his task.

- The batch processing mode is designed to maximize utilization of the system's resources while preventing conflicts in resource use. Batch jobs may be submitted to the batch job stream through the central site card reader, from an on-line terminal, or from a remote site via the remote processing mode.

- The remote batch processing mode provides flexible communication between CP-6 and a variety of remote terminals. Remote terminals can range from a simple card reader and line printer combination to another large-scale computer system with an assortment of peripheral devices.

- Network processing facilities allow the distribution of communications processing into a network of Level 6 minicomputers with their associated software. Networks of Level 6 minicomputers may be geographically distributed and connection may be to several CP-6 host computers as well as to a variety of terminals.

- Transaction processing facilities allow users at remote terminals to simultaneously enter business transactions utilizing a common database. The transaction processing system provides features for communication network definition, transaction queueing, and journalization.

- The <u>real-time processing</u> facilities provide capabilities to implement sensor-based, real-time applications. Portions of the CP-6 real-time capability are distributed to Level 6 processors. Data reduction and analysis tasks more suited to a large system can be performed on the host Level 66 system, while the actual sensor-based applications are being performed on one or more Level 6 systems.

<u>CP-6 Features</u>

The list below outlines those features, facilities or principles which give CP-6 a special uniqueness among large-scale, full-function operating systems.

- Program compatibility in all concurrent operating modes. Programs may be run in any mode without modificaton (subject to very minor restrictions).

- Single, central file management facility within the system. Files are completely compatible and shareable across operating modes and language processors, eliminating the need for file conversions. The file system is easy to use, provides access security, dynamically allocates secondary storage file space, and is accessed compatibly with devices such as card readers, line printers, tape drives, and user terminals. Optionally, files may be updated concurrently by separate programs.

- Event driven, priority-adjustable scheduler integrated with virtual memory management.

- Highly interactive response at time-sharing terminals which is nearly independent of system load.

- Multiprocessing with up to four central processing units per system.

- An excellent remote processing system, including dynamic workstation definition, concurrent master and slave operation, and network support. The system supports any HASP-360/20 protocol IRBT as well as remote batch terminals. The remote processing system is integrated with spooling as its buffering mechanism.

- Ease of use for the casual user in both batch and time-sharing modes. A simple yet comprehensive language exists for control of time-sharing and batch jobs.

- System default conventions that simplify programming and the specification of job control commands.

- Time-sharing access to all devices. Devices are spooled if appropriate.

- Easy-to-use system I/O services permitting access to standard as well as special devices.

- Time-sharing access to almost all programs.

- Sophisticated debugger which has a simple yet comprehensive set of functions suitable for debugging programs of varying source languages. The debugger may be used interactively or in the batch mode.

- Terminal personality including typeahead, echoplex, support of virtually any terminal -- all ASCII TTY compatibles, CRTs, TTY Model 40, Memorex, Execuport. Special handling of tabs, paper tape, transparent (unconverted) I/O, dynamic timing algorithms may be set for each individual terminal.

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

- Master and slave modes for time-sharing terminals. A single terminal may be the source of program control commands and/or one or more terminals may be acquired as passive devices slaved to a program.

- Services supplied by CP-6 have a low overhead.

- High I/O performance via tree structure file indexes with several forms of I/O caches and program-disassociated buffering.

- Automatic, operator-free crash recovery with complete preservation of current user file updates, retention of batch jobs awaiting processing or waiting to print, and a complete diagnostic analysis of the memory dump. The seriousness of the recovery is determined and an appropriate level of recovery is automatically chosen.

- Hardware diagnostics available for time-sharing terminals at local and remote sites.

- Compatibility with other operating systems through ANS labeled tape in either ASCII or EBCDIC and the HASP communication protocol.

- Excellent protection and security of programs and files.
- Shared reentrant system processors which may be user-supplied and may be dynamically added or changed during system operation. These include language processors such as APL and BASIC, public libraries such as that for FORTRAN, and user-written programs.
- Common command processor for interactive and batch users as well as provision for installation-specific, specialized command processors.
- A modern, extensive, and secure data base management system, I-D-S/II, interfaced at the source language level to COBOL-74 and APL. An interface is also available for ANS FORTRAN, PL/I, PL-6, and assembly language programs.
- Ghost jobs for a variety of system and user tasks.
- A comprehensive accounting system including a dynamic charge rate structure and budget accouting.
- An integrated performance monitor for measuring and tuning system performance.
- Simple to use, high-speed system definition and initialization process.
- Completely relocatable and symbolic system on-site system maintenance.
- Most of implementaton code written in a high-level modular language (PL-6).
- Small staff requirement for installation and system support.
- A transaction processing system which includes features for for assuring transaction completion through journalling and recovery services and through increased processing time in response to increased load.
- A transaction formatting system and language which disassociates terminal I/O from the application program.
- Ability to let the system run in an unattended mode (i.e., without the presence of the operator).

ORGANIZATION OF THIS MANUAL

Chapter 2 provides an overview of the operating system. Chapters 3 through 9 discuss elements of the system which are common to all modes of operation. Chapters 10 through 15 each describe those features of the system which deal exclusively with the time-sharing, batch processing, remote batch processing, network processing, transaction processing, and real-time modes of operaton, respectively. Chapters 16 through 22 describe those features of the system which are of interest to the system manager, system programmer, and computer operator. Appendix A describes the hardware configurations supported by the first release of CP-6. Appendix B lists the CP-6 user software manuals.

SECTION II

THE OPERATING SYSTEM

OVERVIEW

The CP-6 operating system consists of a monitor and a number of associated processors. The monitor provides two basic functions:

- Control of the entire system operation, making efficient use of system resource and providing good system response.
- Services to the user that enhance the capabilities of the hardware, providing to each user an extended machine enclosed in an envelope of security.

The associated processors provide specific functions such as compilation, execution, and debugging. All the processors available for a CP-6 system are listed and catagorized in Figure 2-1. These processors are briefly described in Section 3, and some are mentioned throughout this manual.

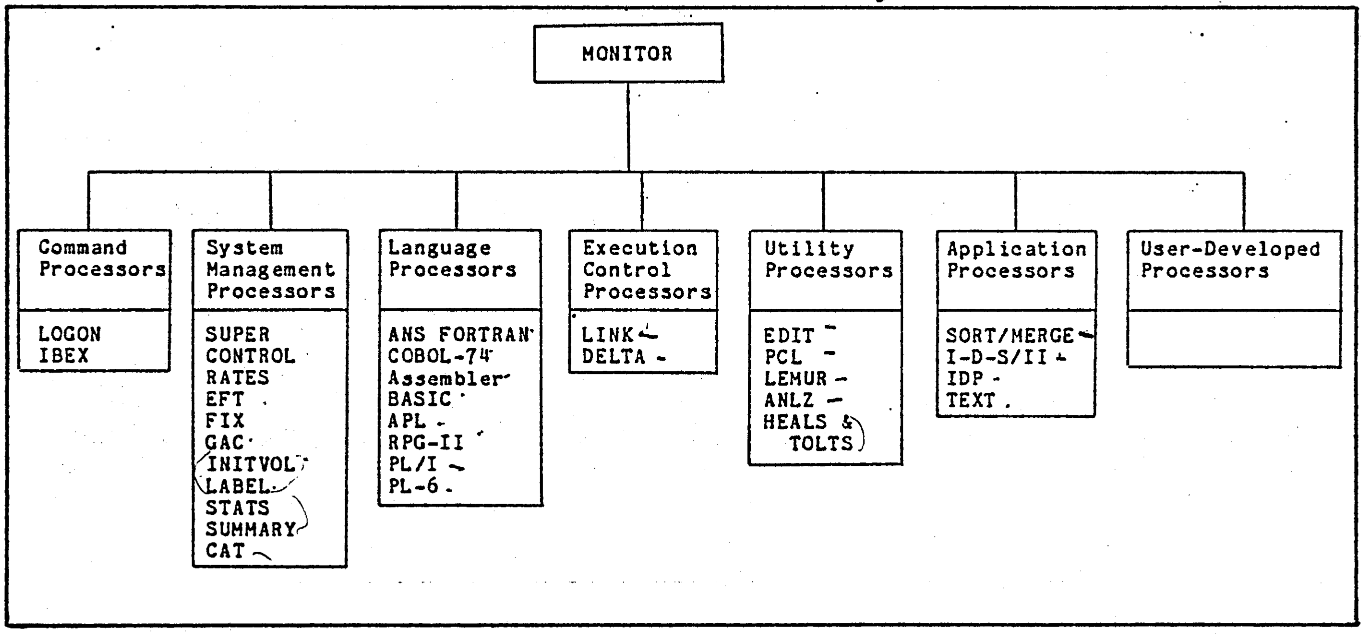

Figure 2-1. CP-6 Operating System

Through the process of system definition, the system manager defines the operating system required for a particular installation. As the requirements of an installation increase, the operating system can easily be enlarged, modified, or updated, because adaptability is inherent in the system design. User programs and the standard system processors are stored, cataloged and referenced in the same manner, and are written using the same services for communicating with the monitor.

The operating system is self-contained, requiring operator intervention only under exceptional circumstances, thereby ensuring faster throughput and a minimization of human error. For the most part, the operator need only perform routine tasks, such as loading and unloading tape reels.

Work is performed on the CP-6 system through a combination of the CP-6 processors and user-developed programs. Each unit of work is packaged together as a "job". There are four kinds of jobs in CP-6 that meet different user (and system) requirements: batch jobs, on-line jobs, ghost jobs, and transaction processing jobs.

Batch Jobs

The system knows the entire control stream and resource requirements of a batch job before the job is put into execution. CP-6 schedules batch jobs to optimize the use of resources. As a general rule, batch jobs are disconnnected from human interaction and output is not delivered until the completion of the job. Errors or abnormal conditions cause the remainder of the job to be discarded. Batch jobs can be submitted from a central site card reader, through an on-line terminal, or through a remote processing terminal.

On-line Jobs

An on-line job receives its control stream from the user at a time-sharing terminal. Resource requirements are not apparent to CP-6 in advance, and are acquired as needed and when available. The user interactively handles unexpected occurences. An on-line job can do everything a batch job can do except pre-allocate resources.

Ghost Jobs

Ghost jobs are not connected to any terminal and have no input control stream. (Many system ghost jobs, however, ask the central site operator for "advice".) Ghost jobs usually provide some special service to the monitor, and their actions are controlled by communication through a file or some other form of internal communication.

Transaction Processing Jobs

Transaction processing jobs make concurrent data base accesses or updates. When a transaction is recieved from a remote terminal, output for the terminal is blocked and queued upon the completion of the transaction processing job.

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. (See Figure 2-2.) The number, types and versions of the programs in an operating system vary according to the exact requirements at a particular installation. Each operating system consists of a selection of monitor routines and processsing programs that are closely integrated for a given range of applications.

The monitor controls or schedules the use of the system resources including CPUs, main memory, secondary storage devices, spooled unit record devices; and terminals of all types. It does this scheduling primarily by deciding when units of user's work should be executed. The monitor also provides extensive services to the users, the installation manager, the computer operator and the hardware support engineers.

Basic hardware control includes routines that field all hardware traps and interrupts and disperse them to the appropriate handlers. It also includes routines and appropriate handlers that schedule or queue and execute all requests for I/O. Since all I/O requests are thus centralized, optimum use can be made of the I/O devices. This centralization also allows for optimum or even experimental use of unique or new features incorporated in future I/O devices.

The CPU scheduler selects the next user or unit of work to be executed on the available CPU(s). This selection algorithm includes some parameters which are modifiable by the installation manager to optimize the system for selected types of jobs.

Job scheduler and resource management routines primarily manage the use of non-shareable resources in an attempt to use the full capacity of the system as specified by the installation manager.

Memory management routines manipulate the hardware tables such that user programs do not require contiguous physical space, can associate and disassociate shared processors and I/O buffers without memory moves, and can contain different program or data areas with varying access restrictions. These routines also provide user services for obtaining and freeing dynamic work spaces.

A spooling facility is provided to help eliminate bottlenecks associated with slow-speed peripheral devices. These routines provide complete buffering between I/O devices and the user's program. Therefore, a user's program never has to wait for an I/O device to complete an action. 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 the spooling routines.

The file management routines provide extensive CP-6 user services:

- Full ANS tape capabilities of label verification, formatting, blocking/deblocking and volume switching
- Six disk file organizations
- Blocking/deblocking
- Data compression
- Security
- Catologing
- Backup and restore
- Archival storage management
- Automatic buffering

Device independent I/O is provided so that programs can be written and debugged without knowing the final disposition of the output or input. When a job is executed it can accept the system default assignments or the user may supply control commands to reassign input/output files or devices.

Job step control provides clean-up services such as closing open files, flagging error conditions, permitting the user to handle abnormal termination, performing step accounting, associating the processor required for the next step, and merging file information into the data control blocks for the next step.

Security measures are designed to prevent unauthorized access to the system, some services of the monitor, programs, and data. Access to the system is controlled by user authorization performed by the installation manager. To access the system, the user must have been authorized and given a user identification. The user identification may include a password which is stored in scrambled form. The entire identification information is not echoed at an on-line terminal during log-on as an added protection. A user's authorization may restrict which monitor services and shared processors are available to the user.

Hardware protection features prevent unauthorized access to memory locations. Users are protected from each other, the monitor protected from the users, users from the monitor, and sections of the monitor from other sections of the monitor. The ring structure of the L6 is used to protect users from one another in communications and real-time processors. A user suspected of attempting unauthorized actions may be aborted by the operator and his authorization to access the system can be dynamically deleted. The memory management routines clear acquired memory to prevent access to data from previous programs.

Levels of access control from account to files, data, and records exist within the file management routines. Accounts and files may restrict access to a specified group of users. Data in a file may be passworded and/or encrypted. Within the I-D-S/II management process, items within each record may be restricted. Secondary storage pages may not be read before being written thus avoiding access to data of previous files. ANS protected or semi-protected controls are provided for tape storage with additional access controls on CP-6 formatted tapes.

During initialization, the installation manager defines the system to reflect the hardware configuration, the number of users, the system features, and the processors to be included. System tables are initialized without requiring installation manager intervention.

Performance tuning parameters exist throughout the system and are used extensively in scheduling tasks. Most of these 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 to provide information to aid tuning efforts.

<u>Recovery routines</u> provide a high level of system availability. Unrecoverable software or hardware failures may result in either a single user abort or a total system crash. Single user aborts remove only the one user from the system. A memory dump is taken and the system then continues. Following a major crash, all the user's files are closed, the accounting log is updated, a memory dump of the critical monitor tables is taken, the monitor is automatically rebooted, and service is then continued. On-line users are asked to log on again and batch jobs that have specified the RERUN option are rerun.

<u>Accounting information</u> is maintained for users and several processors are provided for processing the information. Hooks are provided to allow the installation 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 a separate rate for each individual resource, creating a total charge for the job. Rates can be changed dynamically and applied to a variety of classes of users.

<u>Error detection</u> routines record all hardware errors discovered by the system in an error log file. This file is then available for analysis by the supporting engineers in their preventive maintenance routines or malfunction analysis.

<u>On-line diagnostics</u> and hardware exercisers are available to the support engineers.

<u>Operator communications</u> inform the operator about set-up requirements, device errors that need attention and current state of the batch queue. Users can send and receive operator messages. Several different interactive terminals may be used as operator consoles and different types of messages can be routed to the appropriate consoles.

SECTION III

CP-6 PROCESSORS

STANDARD CP-6 PROCESSORS

The processors that are available for use with CP-6 are briefly described in alphabetical order in this section.

APL

APL is an acronym for A Programming Language, the language invented by Kenneth Iverson. It is an interpretive, problem-solving language. As an interpretive language, APL does not wait until a program is completed to compile it into object code and execute it; instead, APL interprets each line of input as it is entered to produce code that is immediately executed. As a problem-solving language, APL requires minimal computer programming knowledge; a problem is entered into the computer and an answer is received, all in the APL language.

Because APL is powerful, concise, easy to learn, and easy to use, it is widely used by universities, engineers, and statisticians. It also has features that make it attractive for business applications where user interaction and rapid feedback are key issues. One of APL's major strengths is its ability to manipulate vectors and multidimensional arrays as easily as it does scalar values. For example, a matrix addition that might require a number of statements and several loops in other languages can be accomplished as A+B in APL. This type of simplification exemplifies APL's concise power.

ASSEMBLER

A macro assembler is to be provided with CP-6.

BASIC

CP-6 BASIC is a powerful compiler and programming language that is based upon Dartmouth BASIC and represents a superset of the proposed ANS standards. Although BASIC is easy to teach, learn, and use, it is useful for a wide range of applications.

CP-6 BASIC provides many significant enhancements over ANS BASIC. These include:

- The language contains comprehensive set of statements, commands, and supplied functions, an extended MAT package, extensive character string manipulation facilities, and both ASCII and binary file input/output.
- Named data and data files can be shared between successively executed programs and accessed by direct statements.
- The complete working storage environment, including program, named data, and current status can be saved and recalled.
- Debugging operations can be carried out at any time. The user can control BASIC's response to run-time errors.
- Program-flow can be automatically traced, and breakpoints specified that interrupt execution, permitting immediate on-line debugging.
- On-line and batch operations are similar, differing only in default device assignments and error response.
- Most statements can be executed directly, allowing the on-line terminal to be used as a "super" calculator.
- Conformity to CP-6 file conventions allows BASIC to access files created by other CP-6 processors, and to create files that can be used by other CP-6 processors.
- Programs can be sealed, permitting the user to execute (but not modify, copy, or view) the program and associated data.

COBOL

CP-6 COBOL offers a powerful and convenient programming language for implementation of business or commercial applications. COBOL is a standard compiler that conforms to American National Standard COBOL 74. All the required modules have been implemented. It offers extensions in the nucleus and communication modules. I-D-S/II subschema and 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 an object-code compilation unit that can be linked with other compilation units from programs written in COBOL 74 or other source languages to form an executable run unit.

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 such that the user can describe the debugging requirements to DELTA in terms close to the language in which the source program was 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. This allows the user to take immediate action to correct an error. In the batch mode, the user is only restricted to that which can be preplanned.

DELTA facilitates the activities of debugging by allowing the user to:

1. Examine, insert, and modify such program elements as instructions, numeric values, and coded information (i.e., data in all its representations and formats).

2. Control execution, including the insertion of break-points into a program and requests for breaks on changes in elements of data.

3. Trace execution by displaying information at designated points in a program.

4. 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 in to aid debugging at any time, even after a program has been loaded and execution has begun.

EDIT

The EDIT processor is a line-at-a-time context editor for on-line creation, modification, and handling of programs and other bodies of information. All EDIT data is stored in a keyed file structure of sequence numbered, variable length records. This structure permits the user to directly access each line (i.e., record) of data.

EDIT functions are controlled through single line commands supplied by the user. The command language permits insertion, deletion, reordering, and replacement of lines or groups of lines of text. It also permits selective printing, 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 are also provided to allow the user to build, copy, merge, and delete entire files.

FORTRAN

The FORTRAN compiler is compatible with most features of the forthcoming (new) ANS Standard FORTRAN language which includes many extensions to the 1966 ANS FORTRAN Standard Language. It is operable under CP-6 as a shared processor, offering services to both the batch user and the on-line user. The user may request, via an option, that the compiler produce either object module output or program execution (LOAD and GO).

Features of the ANS FORTRAN compiler include

- Compressed input/output capability.
- Addition of INCLUDE (system) capability.
- Conversational characteristics for time-sharing.
- New ANS FORTRAN compatibility.
    - CHARACTER variables.
    - Expanded READ/WRITE capabilities.
    - OPEN and CLOSE statements.

GENMD

The GENMD processor permits on-line, batch, and ghost users to make permanent modifications to existing run units, thereby reducing the number of compilations required to debug a program. GENMD patches are used to modify nonresident elements of the system.

IBEX

The command processor for CP-6 is called Interactive and Batch Executive (IBEX). IBEX commands identify the user job, the tasks to be performed by the job, and the resources required by the job. There are also IBEX commands for functions such as controlling interactive terminal operations. All batch jobs and most interactive sessions require the use of IBEX commands. The language is the same for both modes of processing. (However, not all commands apply to both modes of processing.)

IDP

The Interactive Database Processor (IDP) is a query language which offers the capability for on-line retrieval and display of data maintained within I-D-S/II data bases. The IDP query language consists of key words and operators that can be combined with data base item names and literals to form meaningful statements.

IDP is very easy to use. The IDP user need not have any knowledge of programming nor of I-D-S/II. All the user needs to know are the commands and the names of the items contained in the data base. The IDP language is completely free form. Query sessions are conducted on-line. Any syntax errors are flagged, a diagnostic is issued, and the user is given the opportunity to correct the syntax on-line.

I-D-S/II

CP-6 Integrated Data Store (I-D-S/II) is one of the most advanced data base management systems available. I-D-S/II is a standard implementation of the recommendations and specifications of the Conference on Data Systems Languages (CODASYL). I-D-S/II is richer in functionality than CP-V EDMS and offers additional security because of CP-6 system features.

Highlights of I-D-S/II include:

- The Data Definition Language (DDL) conforms closely to the CODASYL Journal of Development specifications. The DDL is free form and natural.

- The Data Manipulation Language (DML) is part of the COBOL 74 language and conforms closely to the ANSI standard.

- Programs can remain independent of many types of changes to the data base data formats.

- An integrated data base may contain up to 68 billion records in integrated or indexed files.
- Multiple batch and on-line user programs may concurrently update the same area of a data base.
- I-D-S/II data structures may be singular, tree, hierarchical or network in any combination.
- A CALL interface and INCLUDE file for workspace definition in PL/I, FORTRAN and assembly language are provided.
- Data base run-time statistics may optionally be turned on and off when testing or debugging new data base application programs.
- The data base manager is a secure shared library in CP-6 and may not be accessed by the user.
- Data base files can only be accessed by the I-D-S/II data base manager.
- Subschema generation and validation allows the data base administrator to restrict the access of user programs to a subset of the schema specified in the DDL.
- Using privacy locks and keys, I-D-S/II users can limit access to information in the data base down to the data item level.

LEMUR

LEMUR (Library Editor and Maintenance Utility Routine) is a processor available in both on-line and batch modes. It builds library files from object files. It 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.

LINK

The LINK processor controls loading and execution of programs. It accepts object units (which are the output of compilers or assemblers) as input and resolves any linkages between them, producing run units as its output. LINK 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.

PERIPHERAL CONVERSION LANGUAGE

The Peripheral Conversion Language (PCL) is a utility processor for operation in the batch or on-line environment. It provides for information movement among card devices, line printers, on-line terminals, magnetic tape devices, and disk packs. PCL is available in both the on-line and batch modes. The command language provides for single or multiple file transfers with options for selecting, sequencing, formatting, and converting data records. Additional file maintenance and utility commands are provided.

PL/1

PL/1 is a powerful block structured language that conforms to ANS standards. Local and global variables and procedures are supported. Significant features include 16 distinct types of data formats, structure variables, dynamic allocation of array and scalar variables, powerful string heading capability, and comprehensive conditional testing mechanisms.

RATES

The RATES processor allows the system manager to set relative charge weights on the utilization of system services. Specific items for which charge weights may be assigned are currently being selected.

RPG II

RPG (Report Program Generator) II is a convenient means of preparing reports from information available in computer-readable forms, such as punched cards, magnetic tape, and magnetic disks. In addition, it is a means of establishing and updating files of information, usually in conjunction with preparation of reports.

RPG II provides its capabilities through generation (compilation) of object programs, each of which is tailored to produce a different set of reporting results and/or file processing desired by the user. The RPG object programs are capable of accepting input data, retrieving data from existing files, performing calculations, changing formats of data, updating existing files, creating new files, comparing data values to one another and to specified constants to determine appropriate handling, using user-defined processing subroutines, using system library subroutines, and printing reports derived from the input and file data.

RPG II has several advantages over the more traditional method of writing object programs in a symbolic programming language. The RPG language is oriented toward the user's problem, describing reporting requirements, rather than toward the mechanics and manipulations of computer usage. The language and specification techniques are easily learned. A user can become proficient in RPG II after writing only a few programs, whereas an equal facility in symbolic programming would require considerable experience.

SORT/MERGE

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

1. Rearranging (sorting) records of a file to a specified order.

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

SORT or MERGE may be run as a single job step or as a subroutine called by another executing task.

STATS

The STATS processor displays and collects performance data on a running system and produces snapshot files to be displayed by the report generator Summary.

SUMMARY

The Summary processor provides a global view of system performance by formatting and displaying the statistical data collected by STATS.

SUPER

SUPER gives the system manager control over the entry of users and the privileges extended to users. Through the use of SUPER commands, the system manager 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 used to create and delete remote processing workstations.

TEXT

TEXT is a word processing language specifically designed to simplify and accelerate textual communication. The TEXT commands are logical and simple, and can be learned in a few hours. TEXT features include:

- Editing may be done by context or by line number, providing maximum efficiency in error correction.
- Documents can be printed in a variety of spacings and formats, including multiple columns.
- The user can specify headings and footings that print on designated pages, with automatic page numbering.
- Indexes and tables of content can be compiled automatically.
- TEXT is compatible with other CP-6 processors.
- Documents can be accessed, printed, and indexed in the batch mode; the user may reassign input and output DCBs to specified devices or files.
- Names, addresses, and other personalized information may be automatically inserted into form letters or business proposals.
- TEXT can arrange data into complex tables; columns and rows within a table structure can be formatted to obtain a variety of appearances.

PROGRAMS IN ACCOUNT Y

This CP-6 account contains a number of unsupported programs that were, for the most part, created in the process of developing CP-6. One of these programs (HELP) provides information about all of the other programs in account Y. HELP is called by entering HELP.Y in response to an IBEX prompt (!). As soon as HELP is entered, it prints NEXT: on the terminal. If the user responds Y, HELP then types a brief description on the terminal, including a definition of the HELP commands.

Two of the HELP commands are L (list) and H (help). If the user enters the command L, HELP lists the names of all programs in account Y. If the user enters the command H, HELP prompts with NAME= whereupon the user enters the name of the program for which information is desired. Other HELP commands provide additional information about account Y programs.

OTHER PROGRAMS

The Software Library Distribution Center that distributes CP-6 contains a large number of useful programs that are not supported by the CP-6 staff. These programs can be ordered from the library for execution under CP-6. This set of programs is listed in the Honeywell Software Catalog.

Most of these programs are contributed by CP-6 users and are supported by the users.

SECTION IV

SYSTEM CONTROL

This section discusses two CP-6 internal control topics, scheduling and multiprocessing.

SCHEDULING

The most vital part of a multi-use operating system is the scheduler, the module whose primary responsibility is allocating the central processor (as a resource) to various tasks. The scheduler is critical in providing fast 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 the 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 into 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 during execution.

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 the 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 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, allowing maximum utilization of both I/O devices and the basic 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 - thereby allowing 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
- Opening or closing of a file
- Completion of a time quantum
- Addition of a real-time task
- Completion of terminal input

Upon being notified of the occurrence of one of these events, the scheduler changes the priority of the associated job. To facilitate the changing of job priorities, the scheduler uses an event-transition technique. This technique can be viewed as a matrix for which one coordinate represents all possible events that can occur and the second represents the status queues. Any intersection defined by the occurring event and the current state of the associated job determines the new priority and the new queue. Because the executing programs and the environment continually apprise the scheduler of their requirements and of the availability of resources, the scheduler can efficiently and effectively optimize the entire system. Dynamic system tuning is a major factor in making CP-6 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 which may be 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 partition and all on-line users.

- QMIN is the amount of processor time guaranteed a job unless pre-empted by a <u>critical</u> real-time 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 the QMIN time.

<u>MULTIPROCESSING</u>

The multiprocessing facility of CP-6 is an extension to the basic mono-processing system wherein one or more additional processors are available as computer peripherals, each with the ability to access all of physical memory. Sharing of physical memory by all processors is fundamental to multiprocessing. It provides the means of communication between processors and allows each processor accessibility to all users.

Only one processor, called the primary processor, may execute the entire body of code comprising the CP-6 monitor. All other processors, called secondary processors, operate as computer peripherals for use by the primary processor. The secondary processors execute all slave mode user compute tasks and those portions of CP-6 monitor service procedure that do not require interaction with stored central monitor tables. Once a task begins execution on a secondary processor, it continues until either a quantum end condition occurs or a need arises to perform a portion of monitor service that requires access to central tables. Monitor services that do not require such access are completed on the secondary processor. When a secondary processor is no longer able to continue execution of its assigned task, it communicates this condition via data and flags and interrupts the primary processor.

Upon notification of the idle condition of the secondary, the primary CPU suspends the user previously assigned and attempts to find a new user for the secondary. In the event that no suitable user is found, the primary will attempt to schedule tasks for each idle slave upon each subsequent entry of the scheduler (e.g., for I/O interrupt, timer runout fault, and monitor service completion).

SECTION V

USER-DEVELOPED PROGRAMS

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 in the time-sharing mode or is punched onto cards.

2. The program is assembled or compiled by calling the appropriate assembler or compiler. The command for calling this processor is the same in the batch and time-sharing modes. Incremental compilers such as BASIC and interpreters such as APL eliminate some or all of the following processing steps, including them (or their equivalents) automatically -- i.e., without explicit user commands. For processors not falling into one of the two categories just discussed, the output of the assembly or compilation is a relocatable object unit.

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. (This is discussed in greater detail below.)

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

Users may execute a program under the control of the debugging processor DELTA to facilitate the location and correction of errors.

Some of the more specific details of user-developed programs are given in the following section.

LOAD AND EXECUTION

The LINK processor controls loading and execution of programs. It accepts object units (which are the output of compilers or assemblers) as input and resolves any linkages between them, producing run units as its output. LINK 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.

Program Overlays

An overlaid program is a tree-structured program that has only one node (the root node) resident in main memory permanently. 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 house the entire program.

If a 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 relationship of the nodes that constitute an overlay program can be represented graphically by means of a tree diagram, as in Figure 5-1 below. The horizontal coordinate of the diagram denotes increasing main storage (address) allocation, from left to right. The vertical coordinate denotes overlays. The leftmost node, or "root", is that portion of the program that resides in main storage throughout program execution. A "path" of an overlay consists of those nodes that may occupy main storage at the same time. The portion of a path that extends from the start of the program (i.e., the root) to a given node is termed the "backward path" of that node.

```
LINK A(C(D)(E))(B(G)(F)) ON SAMPLETREE
```

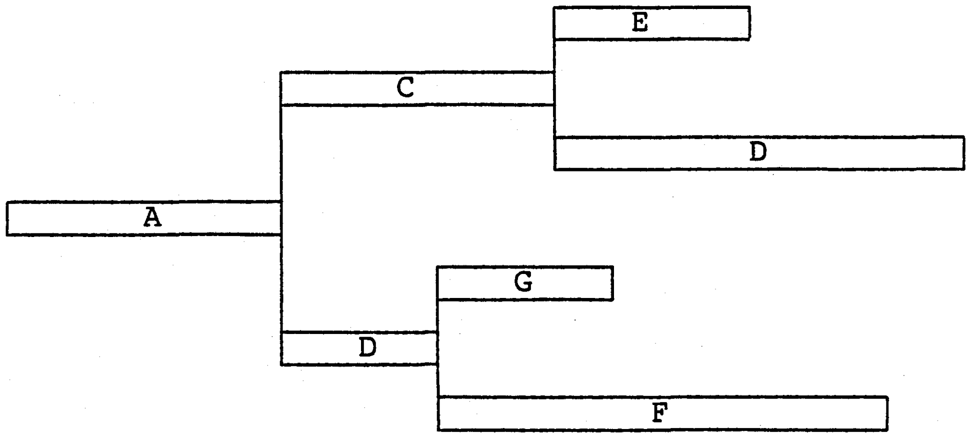

Figure 5-1. Sample Tree Structure

The example in Figure 5-1 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 file are saved in disk storage and, 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, below) must be done in a part of the backward path common to both. Although the tree below is singular, most actual overlaid programs consist of two parallel trees, one for data and one for program. Both are fetched by a single node load call.

RELATED UTILITY PROCESSORS

There are two processors, PCL and EDIT, which are particularly useful in the creation and manipulation of files.

Peripheral Conversion Language

The Peripheral Conversion Language (PCL) is a utility processor designed for working with files in a batch or time-sharing environment. It provides for information movement among card devices, line printers, on-line terminals, magnetic tape devices, and disk storage.

The command language provides for single or multiple file transfers with options for selection, sequencing, formatting, and conversion of data records. Additional file maintenance and utility commands are provided. The PCL commands are summarized in Table 5-1.

EDIT

EDIT is a line-at-a-time context editor for creation, modification, and manipulation of files of text.

All EDIT data is stored on disk in keyed files of sequence-numbered variable-length records, which permits EDIT to directly access each line or record of data. EDIT functions are controlled via single-line commands from the user. The command language provides for the following:

1. Creating a sequenced text file.

2. Inserting, deleting, reordering, and replacing lines or groups of records within a file.

Table 5-1. PCL Command Summary

Command	Function
COPY	Copies file(s) between devices or between public storage and devices.
COPYALL	Copies files from labeled tape or disk pack to any output device.
COPYSTD	Copies a control file and all files named within the file.
DELETE	Deletes specified file(s).
DELETEALL	Deletes all files or a specified range of files in the job's account.
END	Returns control to the monitor.
LIST	Lists files names and, optionally, attributes from the account directory, tape, or disk pack.
REMOVE	Removes a magnetic tape or disk pack.
REVIEW	Lists files in the job's account and waits for a user response after listing each file name to allow the option of deleting the file.
REW	Rewinds a tape reel.
SPE LT	Spaces to the end of the last file on labeled tape.
SPF FT	Positions free form tape forward or backward a designated number of files.
TABS	Set table values for tab expansion.
WEOF	Writes an end-of-file on the current output device.

3. Selective printing and renumbering.

4. Merging part of one file into another.

5. Selecting records for intrarecord editing based on the presence or absence of specified character strings.

6. Context editing operations that allow deleting, moving and substituting character strings within a previously selected set of records.

7. Maintaining files (allowing the user to build, copy, and delete whole files of text lines).

Files may be edited in the user's account (i.e., the log-on account) or under accounts to which the user has been granted write access by the file creator. A user may copy his own files or those to which he has read access. Under the rules of CP-6 file access, a file may not be created (i.e., built or copied to) under an account number different from that used for log-on unless a very high privilege level is associated with the log-on account.

An example of the use of the EDIT processor to create a file is given in Figure 5-1. (Output from CP-6 is underscored. User input is not underscored.) All of the EDIT commands are summarized in Table 5-2.

```
!BUILD PRIME
```

The user wants to create a file called PRIME. (EDIT is called implicitly by the BUILD command.)

```
1.000   10   REM     GENERATE PRIMES GR THAN #
```

EDIT prompts for input by printing 1.000. The user types the first line, then types lines 2-10 in response to more prompts by EDIT.

```
2.000   20   P=1
3.000   30   P=P+4,S=0
4.000   40   FOR I = 5 TO SQR(P) + 1 STEP 2
5.000   50   Q=INT(P/I)
6.000   60   IF Q*I=P THEN 80
7.000   70   PRINT P''TAB(0)
8.000   80   IF S=1 THEN 30
9.000   90   S=1, P=P+2
10.000 100   GOTO 40
```

The user types a carriage return immediately following the prompt for line 11.000 to indicate end-of-file, that is, that the last line of the file has been entered.

Figure 5-2. Sample EDIT Session

Table 5-2. EDIT Command Summary

Command	Function
A	Aligns character strings to specified positions within a record.
AD	Allows addition of text to the end of a record or selected records.
BP	Sets the blank preservation mode. When "on", all strings of blanks are preserved during intrarecord operations. When "off", blank strings are compressed to a single blank or expanded as required to retain column alignment of nonblank fields. The default mode is "off".
BUILD	Enables the user to create a new file.
CM	Causes EDIT to insert commentary into specified columns of each successive record beginning at a specified record.
COPY	Copies one file to another file.
CR	Controls the inclusion of the carriage return character at the end of each record in the user's output file.
CRPT	Controls data encryption of information in the file.
CT	Causes EDIT to type the record up to a specified column and then to insert commentary (given by the user) into specified columns of each successive record beginning at the specified record.
D	Locates a given occurrence of an indicated string, between columns specified by an SE, SS, or ST command, and deletes it.
DE	Deletes all records whose sequence numbers lie in a specified range.
DELETE	Deletes the file specified from the log-on account.
E	Starts at a column occupied by the first character of a given occurrence of a specified string or column and overwrites with another string.

Table 5-2. EDIT Command Summary (cont.)

Command	Function
ECHO	Causes commands executed via the EDIT XEQ command to be displayed.
EDIT	Opens a file to be edited and enters the record editing mode.
END	Closes all active files and returns control to the control command processor. This command is equivalent to the X command.
F	Starts after the last character of a given occurrence of a specified string or column and inserts another string, pushing everything from this column right as required to make room.
FD	Searches for the presence or absence of specified strings between specified columns in a specified range of records. If the string is found, the record containing it is deleted from the file.
FS	Searches for the presence or absence of the specified string between specified columns in a specified range of records. Each time the string is found, the sequence number of the record is printed.
FT	Searches for the presence or absence of the specified string between specified columns in a specified range of records. Each time the string is found, the sequence number and the contents of the record are printed.
IN	Inserts new records into a file starting at a specified record. EDIT prompts the user with the sequence number of each record to be inserted.
IP	Inserts new records into a file only if a record with the same sequence number doesn't already exist.
IS	Inserts new records into a file starting at a specified record. EDIT does not prompt with sequence numbers of the records to be inserted.

Table 5-2. EDIT Command Summary (cont.)

Command	Function
JU	Causes the SS or ST command to jump to the specified record and then continues stepping from that point.
L	Shifts portions of the record left the number of positions indicated.
MD	Moves records within a file from one range to another range. The original records are deleted.
MERGE	Merges records from one file into another file.
MK	Performs the same function as MD except that the original records are not deleted as they are moved.
NO	Specifies that no editing is to be performed on the current active line.
O	Starts at the column occupied by the first character of a given occurrence of a specified string or column and overwrites with another string.
P	Starts before the first character of a given occurrence of a specified string or column and inserts another string, pushing characters of the first string to the right as required to make room.
R	Shifts portions of the record right the number of positions indicated.
RF	Causes the current setting of the blank preservation mode ("on" or "off") to be reversed temporarily (for the current line only).
RN	Renumbers a specified record.
RP	Sets the record size preservation mode. When ON, the size of the edited records is not changed. Trailing blanks are not deleted. When OFF, records are shortened or lengthened as necessary by the editing process. Trailing blanks are deleted. The default mode is OFF.

Table 5-2. EDIT Command Summary (cont.)

Command	Function
RR	Types the currently active record and allows interactive terminal input type editing on it.
S	Locates a specified string between columns specified by an SE, SS, or ST command and replaces it with another string.
SE	Causes EDIT to accept successive lines of intrarecord commands to be applied to records selected by a range of line numbers and the presence or absence of certain character strings within selected columns.
SS	Causes EDIT to start at a specified record and proceed to each record in succession, accepting one line of intrarecord commands to update the current record.
ST	Causes EDIT to start at a specified record and proceed to each record in succession, accepting one line of intrarecord commands to update the current record. The sequence number and contents of each record are typed prior to accepting a command.
TA	Causes EDIT to set or reset the terminal tab stops to settings appropriate for a specified language processor.
TABX	Controls the tab expansion mode. If on, tab characters are replaced with blanks when read from a file and blanks are replaced with tab characters when written. If off, tab characters are not changed.
TC	Types the sequence numbers and the contents of specified columns of one or more records beginning at a specified record. Any nonblank strings within the columns typed are shifted to the left to compress each blank string to a single blank.
TS	Types the contents of the record currently open for editing under control of an SE, SS, or ST command or types the contents of specified columns of one or more records beginning at a specified record.

Table 5-2. EDIT Command Summary (cont.)

Command	Function
TX	Types the sequence number and contents of those records within the edit range (set by SE, SS, or ST commands) which have been changed by the preceding intrarecord command(s).
TY	Types the sequence number and contents of the record currently open for editing under control of an SE, SS, or ST command or types the sequence numbers and the contents of specified columns of one or more records beginning at a specified record.
X	Closes all active files and returns control to the executive language (IBEX). This command is equivalent to the END command.
XEQ	Causes EDIT to execute commands from a specified file.

SECTION VI

FILES AND DEVICE INDEPENDENT I/O

INTRODUCTION

CP-6 provides all I/O services through a set of services that are common to all modes of system operation. Programs may be written without the need for explicit knowledge of the file or device to which I/O will actually take place. Selection of the file or device can be made internally via a system service (M$OPEN) or externally via an IBEX control command (SET or LDEV). In addition, a set of default device assignments may be used which make appropriate device selections for batch or on-line jobs.

All requests for I/O services specify a data control block (DCB) name for use 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 6-1 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 the CALL M$name (FPT_name) mechanism.

2. The DCB is connected to one of several types of I/O facilities via M$DCB, !SET, !LDEV, or M$OPEN.

   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. Operational label - a label that has been assigned a "functional" connection (e.g., "the place where all listings go") to a spooling file or to a user terminal as appropriate for on-line or batch.

   d. Device type - a name which is connected to a spooling file, the user terminal, or a specific device (e.g., a specific tape drive with a foreign tape mounted, which has been defined to be a resource).

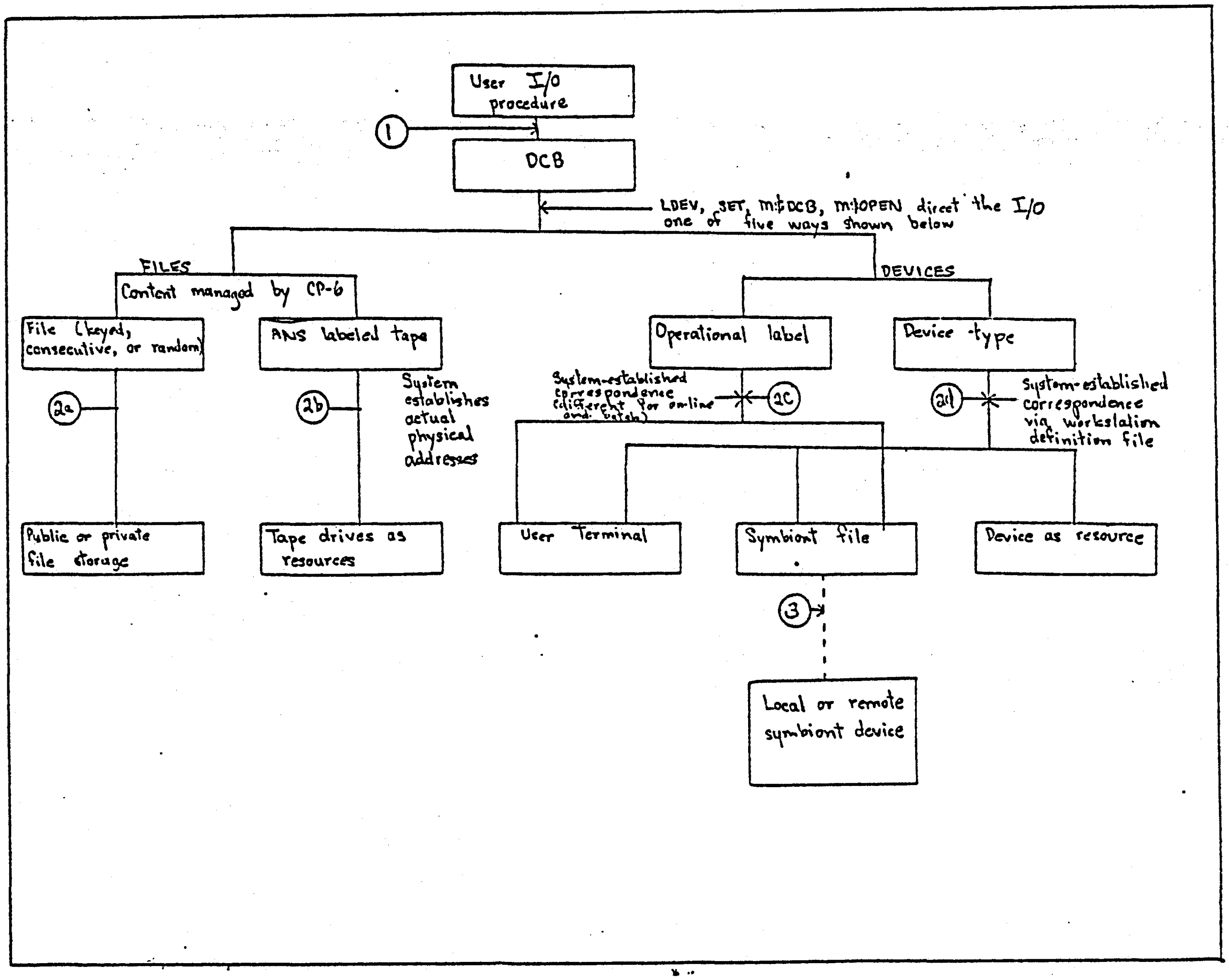
User I/O procedure
1
DCB
LDEV, SET, MDCB, MOPEN direct the I/O one of five ways shown below
FILES
Content managed by CP-6
File (keyed, consecutive, or random)
ANS labeled tape
2a
2b
System establishes actual physical addresses
Public or private file storage
Tape drives as resources
DEVICES
Operational label
Device type
System-established correspondence (different for online and batch)
2c
2d
System-established correspondence via workstation definition file
User Terminal
Symbiont file
Device as resource
3
Local or remote symbiont device

Figure 6-1. Connection Established for Performing I/O

3. The contents of a spooling file are copied to a local or remote device (line printer, card punch, etc.) or are built from such a device (e.g., card reader). Spooling file destinations are established by default conventions or by the IBEX LDEV command.

For convenience, a number of DCBs are available that have default assignments to the operational label of the same name. For example, LO (listing output) and SI (symbolic input) are the assignments of the M$LO and M$SI DCBs. 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.

PACK SETS

All random access devices known to CP-6 are initialized by INITVOL or by system initialization. All initialized random access devices, whether dismountable or non-dismountable, belong to some _pack set_, a group of one or more devices. A pack set is identified by its _pack set name_, a one to six character name common to all devices in the set.

Each member of the pack set also has a unique _serial number_, used by CP-6 when mounting a pack set. External usage of serial numbers is restricted to the system manager or operator for communication with CP-6 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.

The set of devices containing the system area is called the _pack set name zero_ (PSN0) set. PSN0 is also the default location of files created in the system account (:SYS) and user temporary files.

Spindles not in the PSN0 set are in the _managed spindle pool_. Pack sets mounted in the managed spindle pool may be accessed in either a shared or exclusive mode. Shared pack sets may be accessed by many users concurrently; exclusive pack sets may be accessed only by the user for whom they are mounted. The shared and exclusive properties of a pack set are determined by its user's mode of access. There is no difference in the internal format of shared and exclusive sets and a pack set may be accessed interchangeably regardless of how it was accessed previously.

The word public means that a pack set has been mounted as public and the accounts on it have been entered into the system account directory. The pack set name is entered into the system catalog, along with serial numbers and device types. This catalog is a file created by the system manager. Any public and private volume sets required by a batch job must be mounted or reserved prior to the job's selection for execution.

FILES

A general understanding of files and the way that the monitor deals with them will help the user to obtain the high level of performance available.

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 CP-6, 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 were being accessed directly.

On each pack set, the monitor maintains a directory of accounts having files maintained between jobs. This is called an Account Directory, and contains, with each account number, an address of a directory of files (termed a File Directory) for that account. A File Directory contains, with each file name, an address of a table containing file attributes and location for that file. The table is called a File Information Table. To summarize, each pack set has a single Account Directory which, in turn, points to a File Directory for each account. Each File Directory, in turn, points to a File Information Table (FIT) for each file.

Each file directory and each file has information associated with it (in the FIT) controlling who may access the directory or the file and how it may be accessed. This information can include a list of which accounts may perform certain controllable operations. To access a file, a user must be running under an account which is authorized to access the file's account and the file and must provide the proper password (if one exists). In addition to access control information, the FIT also contains the various file dates, special installation information, and file type.

File Identification

In CP-6, files are identified according to a standard format. The identification is assigned to the file by the user when the file is created. The format for a file identifier (fid) is one of the following:

```
name
name.account
name..password
name.account.password
name.account..setname
name.account.password.setname
```

where

name is the name of the file and may have a maximum of 31 characters. (Certain processors in CP-6 may have shorter length restrictions.)

account is the account number of the file and may have a maximum of eight characters. A user normally may not create a file in any account other than the one under which he is running. He may not execute, read, or modify a file in another account unless he has been given the appropriate access rights to that file.

password is the password for the file and may have a maximum of eight characters. A password is useful in preventing unauthorized users from accessing data in the file.

setname is the pack set name of from one to six characters in length.

The various combinations for file identifiers are determined by which data are required.

name is always required.

account is required only when the account is different from the current user's account.

password is required only when the file has a password associated with it.

setname is required only to access privately mounted pack sets.

File Function and Disposition

A file may be opened for one of four functions: input, output, update, or journal. There are two possible dispositions for a file -- either to save it or to release it. The specification of whether to save or release a file may be made when the file is opened and/or when the file is closed. If a file is opened in the save mode, the user can have a change of mind and close the file in the release mode. However, a file that is opened in the release mode may not be closed in the save mode. If the disposition of a file is not specified, the default (save or release) depends on the function of the file.

A file opened and closed with the CREATE option and with a save disposition becomes a permanent file and storage used is accounted for as permanent granules. A CREATE file with a release disposition is a true scratch file which is never cataloged and storage use is accounted for as temporary granules. The default close at job step or across recovery is release so that open output files are released. (Files associated with the diagnostic output DCB (M$DO) are an exception so that diagnostic output is not lost.)

There is a special class of files called star files. These files exist for the duration of a job and are automaticallly released at the end of the job. Therefore they accounted as temporary granules. Star files are guaranteed by the system to be unique for each user. Thus a processor running for different users may have intermediate files of the same name but they will not be confused among the users.

With one option, a user may request opening of an existing file, or 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 exists, or creation of a new file if no file of the same name exists.

File Organization

The information in a file may be structured in one of three ways: as a keyed, consecutive, or random file. There are two suborganizations: indexed and relative. 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). Thus file access is simply the way in which a file is being accessed at a particular instance of usage.

KEYED AND INDEXED FILES

Keyed files are those in which each record has an identifying key associated with it. A key consists of a byte string, the first byte of which states 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 may consist of up to 31 bytes plus a count byte.

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 sorted into collating sequence so the file can subsequently be accessed sequentially. The entry contains such information as the key, file-relative disk address of the record, size of the record, and position of the record within the blocking buffer. Thus records in keyed files may be accessed directly or sequentially.

Keyed files have a multilevel index structure to provide fast direct access. A multilevel index structure is a collection of hierarchical levels of index blocks, where the entries in a higher level point to index blocks at the next lower level and the entries in the lowest level (called level 0) point to data records. This is best illustrated by the hypothetical example shown in Figure 6-2 Unless specified otherwise by the user, the multilevel structure is initially built when the file is closed if the file has more than three level 0 index blocks.

This example is computed for a fixed length key when in fact the keys are of variable length and would be more tightly packed. Additionally, the key entries on the higher levels are further compressed to the minimum value required for uniqueness. This example also shows the data in the same sequence as the keys. This need not be true and will not be true after random addition of records.

Each entry in a higher-level index block contains the disk address of an index block at the next lower level, and enough of the first key in that block to differentiate it from the first keys in adjacent blocks.

The multilevel index structure can considerably speed up the direct access of a large keyed file, at only a small cost of secondary storage space. Since the keys are ordered in ascending sequence, at most it would take three index block accesses to locate a data record as shown in the example. Without the higher-level structure, it could take up to 390 index block accesses.

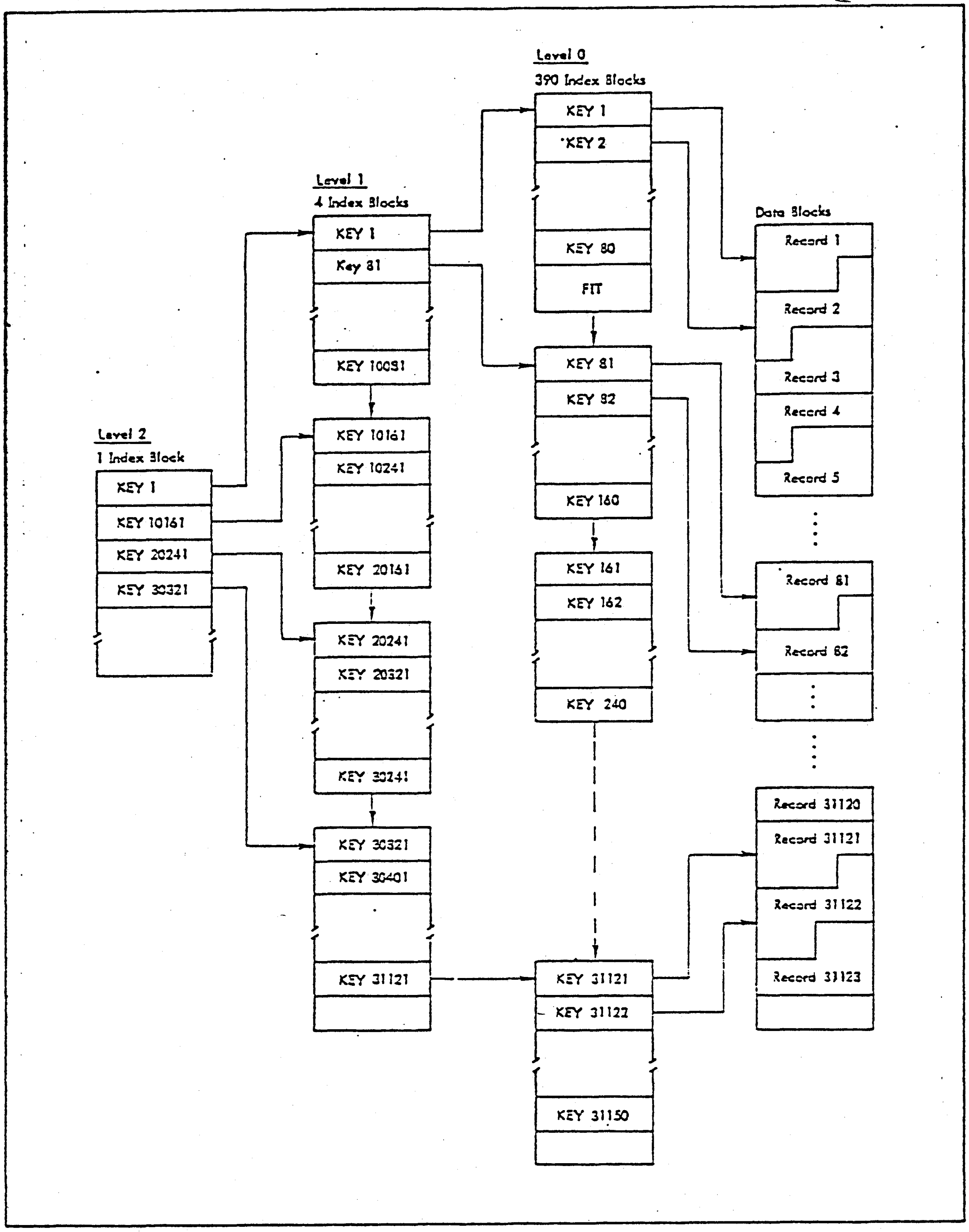

Figure 6-2. Example of Multilevel Index Structure

The user has control over the initial creation of the multilevel index structure and can specify when and if the higher-level structure should be rebuilt. This can be specified by using the NEWX option on the SET control command or the M$OPEN and M$DCB procedures.

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. Secondary index capability is not provided.

CONSECUTIVE AND RELATIVE FILES

Consecutive files are files whose records are organized in a consecutive manner; i.e., the user is not aware of any identifying keys associated with the records. The records may only be accessed sequentially.

The principal benefit of consecutive files to system operation is a reduction in the amount of space required for the files on disk and a consequent reduction in the time required to traverse the files. Also, there is no need to identify each record.

Many positioning operations for consecutive files are performed with user selected system procedures rather than with I/O operations. The positioning is only effected when a data transfer operation is about to take place. At that time, there are three known points in the file that can be used as a starting point: beginning of file, end of file, and the position reflected by the DCB. The starting position chosen is the one that requires the fewest record skips to be made.

Relative organization is implemented as a special case of the consecutive file organization. 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 directly by record position.

RANDOM FILES

Random files provide a basic organization for those users desiring to manage their own files. Random organization differs from keyed and consecutive organization as follows:

1. A random file is simply a collection of logically contiguous granules on the user's pack set. The number of granules is specified at the time the file is created and may be expanded dynamically. If the

requested number of granules is not available, an error code is returned to the user and the file is not opened.

2. The user may specify a relative starting granule number and a byte count with each read or write.

3. Each write consumes the entire specified granule. Bytes not specifically written contain no useful information. The contents of the granule include no system information except the granule stamp. The one-word granule 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.

Thus, the monitor provides allocation of granules, security checks, and normal I/O queuing service and clean-up. The user is responsible for record management.

File Access

Records within a file may be accessed by either of two means, direct or sequential access. The interaction of the type of access used for a given operation and the mode in which the file is opened results in the limitations described below.

Direct access of consecutive file has meaning only for input files and the meaning is that read-ahead is inhibited.

For direct access reading of (IN,UPDATE) keyed files, the key may or may not be specified. If a key is specified, a search is made of the file until the key is found and the record is then read. If the key is not found, an error return is executed. If a key is not specified, the next sequential record is read.

Direct access for writing of output (CREATE) keyed files requires that the keys be specified. The keys do not need to be given in a sorted order. They will be ordered as they are stored.

Direct access for writing update (UPDATE) keyed files may or may not specify the key. If a key is not specified, the write function must have been preceded by a read. If it is, the record just read is updated; if not, an error code is signaled.

Write operations may indicate whether the record is intended to be a new or replaced record. The absence of an option indicates replacement by default.

The DELETE function may be specified. If a key is specified with DELETE, a search of the directory is made to find the specified key. The record is then deleted. If a key is not specified, the DELETE operation must have been preceded by a READ, and the record just read will then be deleted.

Sequential access to input and update (IN, UPDATE) files is the same as for direct access.

Sequential access to create (CREATE) a keyed file requires that a key must be specified and this key must be a new key (i.e., it must not have been used before). If the key has already been used, no information is written and an error code is returned. The keys must be given in a sorted order. For example, if the user writes records with keys A, C, and D, respectively, and then writes a record with key B, the record will not be written and an error return will be executed.

The position record forward and position record backward operations are allowed on both keyed and consecutive files. Error return is made when beginning or end of file condition is encountered. Otherwise, for keyed files, the pointer to the current entry in the master index is decremented or incremented. For consecutive files, a directional count of records to skip from the current position is established. Positioning will not occur until the next read, write, or delete operation.

In sequential access of an update file, a read order issued prior to the first write results in an error return. When reading a keyed file, a key may or may not be specified. If the key is not specified, the next record in sequence is read and the key may be returned to the user if requested.

SIMULTANEOUS FILE ACCESS

The SHARE mode feature extends the use of keyed and random files by permitting simultaneous access to a file by many updaters <u>and</u> many readers. Thus 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 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 and also ensures that readers are provided with the most up-to-date information in response to their requests.

Coordinating and guaranteeing logical integrity of the file (primarily 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. A single example will serve to illustrate this.

Suppose that a file contains records recording a parts inventory - each containing the available number of bolts, washers, nuts, etc., in various sizes. Without any special coordination, the number of any given item can be determined by querying the file even in the face of additions and removals by a concurrent updater. If, however, the application needs to first determine the available number and then remove a quantity from stock, then the record must be locked between the read and the update to preclude the possibility of the stock being taken by another updater.

More elaborate record locking requirements may exist depending on the application. For example, if a fastener must be made up of a bolt, a nut and a lock washer, then these three records must be acquired and locked prior to making the needed updates.

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. It is the responsibility of all users of the service to agree on the meaning of the names - for example the names of the records containing inventory count of nuts, bolts, and washers.

There is also 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 a file is closed, it is no longer a journal and may be accessed as any other consecutive file is accessed.

Record Blocking

The system automatically blocks records for keyed and consecutive files in 1023-word blocks to provide more efficient use of disk space. The user has no knowledge of this blocking and, when reading, will receive the appropriate record within the block and not the entire block.

When updating a keyed file, the user may rewrite a record in a size larger or smaller than the original record size. If necessary, the monitor will allocate additional disk space to accommodate a larger size. A record may also be rewritten in a consecutive file but the original record size is maintained. If the new record is larger, data is lost; if it is smaller, old data remains in the record.

A write with a 0 byte count to a keyed file results in a master index entry for the record with fields in the entry pertaining to disk address, record size, and displacement into the blocking buffer all set to zero. A write with a 0 byte count to a consecutive or random file is ignored.

Special Features of the File System

In addition to the facilities of the file system as outlined above, there are a number of special features which make CP-6 easier to use:

- File Extension - A limited set of system defined DCBs (M$operational label) are provided file extension by the system. Thus the first use of a system output DCB within a job will create a new file and subsequent uses add to the end of the file without explicit action on the part of the using program.
- An option is available to determine the next account in the account directory and the next file in a file directory provided the user has proper authorization.
- Assignments to the M$SI (source input) DCB by on-line jobs are extant for one job step only (which is what the user wants and expects but may forget to do for himself).
- Data compression and automatic decompression may be requested by the user.
- Read-ahead, write-ahead, and I/O cache are implementation details which enhance the performance so that 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, vertical format control, etc.).
- Files have a code which is used by programs to determine the type: APL workspace, various FORTRAN output formats, BASIC program, etc.

- In most cases, files are automatically extended in size as the files grow. Random files are extended by specific request rather than by automatic extension.

File Security

The integrity and security of the file system is critical to ensure safe and reliable operation. The CP-6 file system uses four separate techniques to prevent unauthorized access. The first is discussed under the heading "Granule Access Control". The remaining three are discussed under the heading "File Access Controls".

GRANULE ACCESS CONTROL

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 some source other than the file in question will be returned to the user or left in any of his monitor buffers unless the stamp is verified. This provides a high level of information security for both hardware and software error situations, and also prevents the user from usefully reading any granule which has not been written.

FILE ACCESS CONTROLS

Access to file data is applied at several levels: account, file, and data. There are two types of account access control: creation of new files and access to existing files. There are about eleven types of file access control. Data access control includes password and data encryption.

Account Access Controls

Each account may specify two access controls: permission to create new files in this account or permission to access existent files either for reading or updating. These controls may be one of the following: NONE, ALL, PUBLIC, or a specified list of accounts.

File Access Controls

There are eleven types of access controls for files (e.g., read, update, delete records, knowledge of its existence...).

Each file may specify a combination of these accesses to be permitted to ALL, NONE or PUBLIC. Also, different lists of accounts. (This control information is included within files on ANS labeled that have CP-6 formatted files.)

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 <u>encoded</u> password is entered into the file information table (FIT).

It is possible to request encryption of data in a file of any organization. This process consists of applying, via an exclusive OR operation, successive elements of a sequence of pseudo random numbers to successive groups of four data bytes. The initial element of the sequence is derived from a seed specified 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.

<u>Reliability, Maintainability, and Availability</u>

Reliability, maintainability and availability for the file system are the cornerstone of the design of file management for CP-6. Some of the features are:

- Unique granule stamps provide not only security but maintainability by providing information to aid in the reconstruction of file(s) when disaster strikes.
- Directory granules are linked and the forward and backward links are checked on access.
- At job step and during recovery, the user's DCBs are closed properly and current updates are posted to the files.
- Dual copies are maintained for critical elements - directories, file information tables, and granule allocation tables.
- Pack sets provide a separation of files such 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.

- File space is allocated by extents (groups of granules) rather than by single granule. This is beneficial in two ways. Damage to several granules may be contained to damage of one file rather than several and 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 granules.

- Each open file has a list of associated users by user number so that recovery may close the files properly even if the user's Job Information Table was destroyed.

- File directory descriptors and dates are updated when the file is opened so that recovery need not handle that.

- Finally, if a file was opened and in some way (major disaster) was not closed, the open process will reconstruct the necessary control information.

Efficient File Transfer (EFT)

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

BACKUP

User files are copied to some backup media according to an installation determined schedule. On an individual file basis, the user may specify that a file should or should not be included in the regular backup scheme. Only files thus qualified that have recently been modified are considered for backup. The backup media may be either ANS labeled tape or a dual pack set. In general, the installation will update the dual pack set when a set is dismounted. It may also back the files onto tape during the day in order to protect active files more frequently. Backup may be performed either in the fastest way possible (block access) with no cleanup, or not quite as fast (record by record) with complete cleanup.

RESTORE

The restore function has two applications: restore an entire pack set because of some large disaster, or 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 a little more complicated since various sets of tapes must be restored 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 (stow) media. 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 (the data is deleted) or active (data is retained) status. 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 media to the active file system.

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 use as a transport vehicle. Using ANS tapes, data can be transported between CP-6 and all other systems which support ANS tapes (other Honeywell systems, IBM systems, etc.).

Protection and Security

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

Additionally two mode options allow the installation manager to specify how rigidly the ANS protection features are to be applied:

1. Protective Mode: A mode of tape protection in which only ANS expired tapes 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; tapes mounted as input tapes may not be written; and tapes mounted as other than input tapes must have a write ring.

2. Semi-Protective Mode: A mode of tape protection in which a warning is sent to the operator when output is attempted on an unexpired ANS tape, or when a tape mounted as available for output has no write ring. The operator can authorize the overwriting of the tape or the override of the input and output specification through a key-in. 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.

The user should be aware, however, of the following restrictions on ANS labeled tape processing:

- Tape cataloging is not available, so Generation Data Groups are not applicable.
- Multiple tape sets are processed by serial number.

Tape Formats

For transport purposes, CP-6 supports the EBCDIC formats F, V, U, and S and ASCII formats F, U, and D. (F is fixed length records, U is unformatted, and V and D are variable length records. The V format records size in EBCDIC and D records size in decimal. S format allows logical records to span physical blocks to allow for long records.)

For tape usage within CP-6, the ANS formats have been expanded to allow additional file types. B format provides blocked access and efficient tape usage for the file maintenance processor. K format provides keyed files which can be processed on tape only in sequential access mode. C format provides consecutive file formats. These three special formats include the File Information Table as part of the data (though invisible to the user) so that the files indeed resemble the files from pack sets. Files can be copied from one to the other and retain their personalities.

Blocking, deblocking, and volume switching is performed by the tape file management system and is not the responsibility of the user.

Multifile Tapes

Once a user has opened a file on a multifile tape, no other user may access the tape until the original user has closed and removed the tape. If the rewind option is specified, the tape is rewound. Otherwise, the tape remains at the current position and, if a DCB is opened using tape, one of two actions occurs:

1. On input or update, the tape is scanned forward for the desired file.
2. On output, the tape is positioned to the end of the current file and the new file is written at that position.

DEVICE INPUT/OUTPUT

Devices used for I/O other than file or labeled tape are basically divided into three classes:

1. Interactive Terminals
2. Unit record peripherals (local or remote)
3. Unformatted Devices

A DCB used for I/O by a user program can become connected to these types of devices by assigning a DCB to an operational label or physical device type. Operational labels 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 device streams or to the interactive terminal. A logical device stream may be directed to unit record peripherals with the LDEV command.

A physical device type assignment causes a connection which depends on the device type. Unit record peripheral DCBs are connected to spooling files, while unformatted device DCBs are connected directly to a device of the proper type using criteria such as serial number for a foreign tape on a tape-type device.

Interactive Terminals

Interactive terminals fall into two classes, master and slave, with a great deal of commonality between classes. Slave terminals are acquired by programs as a contention resource (not pre-allocated) if not already acquired or logged on as a master terminal. The terminal may then be used as an I/O device by the program until released. Slave terminals are completely under control of the program.

Master terminals are associated on a one-to-one basis with on-line jobs. In addition to being an available I/O device, the terminal is the control device for the on-line job. JCL commands are read from the terminal. The BREAK key can be used as a program interrupt and the CONTROL Y sequence causes return to the associated command processor. Thus the master terminal is in control of the program. Whenever a time-sharing user logs on, his terminal automatically becomes a master terminal. (See Sections 10 and 13 for more-information on communication terminals and time-sharing terminals.)

Regardless of type, terminals are usable as I/O devices with a great deal of flexibility. Terminals can be operated in echoplex mode or with local printing. Sophisticated editing and tabbing features are provided on input. When operating in echoplex mode, typeahead is allowed with proper sequencing of input and output guaranteed. Tabbing, line breakup to fit device, pagination, and page headers are provided on output. 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 (card readers, card punches, line printers, plotters) is staged via spooling files whether the device is local or remote. This simply means that the entire input job from a card reader is read onto disk before processing and all of the job's output to these devices is stored on disk, normally not being output until the job is complete. (Spooling is discussed in more detail in Section 11.)

The processing of these devices symbiotically provides the following benefits:

- Disconnects program execution from I/O devices
- Smooths peaks and valleys in I/O demand
- Allows multiple programs to output to the same device simultaneously

- Allows grouping of output by form type
- Allows a program to generate several 'streams' of output to one device
- Allows several copies of output to be produced
- Enables on-line use of batch peripherals
- Allows submission of jobs to the batch job queue from on-line terminals
- Allows pre-scanning of job requirements 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 is responsible for the data read or written to the device. No blocking or formatting services are provided.

SECTION VII

THE COMMAND PROCESSOR

The command processor for CP-6 is called Interactive and Batch Executive (IBEX). IBEX is an interface between the user and the operating system and is directed by its various commands. These commands control the construction and execution of programs and provide communication between a program and its environment. The environment includes the monitor, processors (such as EDIT, COBOL, LINK, DELTA), the operator, peripheral devices and, for a time-sharing user, the on-line terminal. The functions performed by several of the IBEX commands may also be performed during program execution via monitor services. (See Section VIII.)

All batch jobs and most interactive sessions require the use of IBEX commands. The language is the same for both modes of processing. (However, not all commands apply to both modes of processing.) IBEX commands are listed in Table 7-1.

CP-6 processors are called by specifying the processor name as a command to IBEX. Similarly, execution of user programs that are in run unit form may be initiated by IBEX simply by specifying the file identification of the file that contains the run unit.

Table 7-1. IBEX Commands

Command Mnemonic	Description
ADJUST	Performs the requested adjustment on any DCB in the user's DCB area (providing a dynamic SET when not at a job step). The format of the command and the options are the same as for the IBEX SET command.
BACKUP	Qualifies a specified file to be saved on the system backup tape during the next backup operation.
BATCH	Enters the specified file(s) into the batch job stream, allowing field and string substitutions first.

Table 7-1. IBEX Commands (cont.)

Command Mnemonic	Description
BUILD	Invokes the EDIT processor to create a file.
BYE	Terminates an on-line session. This command is equivalent to the OFF command.
CANCEL	Cancels previously submitted batch jobs.
COMMENT	Directs error commentary to the specified device or counteracts the preceding DONT COMMENT command.
CONTINUE	Restarts a program that was interrupted with CONTROL Y or that was RESTOREd. This command is equivalent to the GO and PROCEED commands.
COPY	Invokes the PCL processor to copy a file or device input to a specified output file or device.
COUPLE	Establishes a link between the user's terminal and the terminal specified; or puts the user's terminal in a mode such that attempts to couple to it will be accepted.
DEBUG	Sets the FORTRAN debug mode.
DECOUPLE	Releases the coupling of a currently coupled terminal.
DELETE	Invokes the PCL processor to delete the specified file(s).
DESTOW	Removes the qualification to be stowed from a specified file. (See the STOW command.)

Table 7-1. IBEX Commands (cont.)

Command Mnemonic	Description
DISPLAY	Lists the current values of various system parameters.
DONT COMMENT	Stops error commentary output from an assembly or compilation.
DONT COUPLE	Causes attempts to couple to the terminal to be rejected.
DONT DEBUG	Resets the FORTRAN debug mode.
DONT ECHO	Suppresses printing of IBEX commands being executed from a command file.
DONT ERROR	Prevents messages pertaining to peripheral device usage (such as tape) from being output at the user's terminal and prevents the user from exiting from conditions requiring operator intervention.
DONT LIST	Stops listing output from an assembly or compilation.
DONT OUTPUT	Stops the output of an object unit from an assembly or compilation.
DONT SEND	Disallows messages from the machine operator to the user's terminal. Global broadcasts are deferred until IBEX is in control. Also disallows the MESSAGE command.
E	Invokes the EDIT processor.
ECHO	Causes printing of IBEX commands being executed from a command file.

Table 7-1. IBEX Commands (cont.)

Command Mnemonic	Description
END	Terminates an interrupted program. This command is equivalent to the STOP and QUIT commands.
ERASE	Deletes the accumulated line printer output for an on-line user.
ERROR	Allows the user to receive the same messages as the operator when dealing with tapes.
FETCH	Requests the restoration of an inactive stowed file to active status. (See the STOW command.)
GET	Restores the previously saved main memory image. This command is equivalent to the RESTORE command.
GO	Continues processing from the point of interruption. This command is equivalent to the CONTINUE and PROCEED commands.
HELP	Prints general information about IBEX and its commands.
INFORM	Permits the output of a message to a user's terminal when a couple is attempted by another user.
JOB	Controls access to the system in the batch mode.
JOIN	Changes the terminal from one communication group to another.

Table 7-1. IBEX Commands (cont.)

Command Mnemonic	Description
L	Invokes the PCL processor to list file names and, optionally, attributes from the file directory, tape, or private disk pack.
LDEV	Modifies a logical device definition; directs a stream of information.
LIMIT	Establishes job or job step limits for various system resources when in batch mode. Additionally, the command may be utilized to release system resources.
LINK	Invokes the LINK processor. This command is equivalent to the LOAD and LYNX commands.
LIST	Directs the listing output to the specified device, or counteracts the preceding DONT LIST command.
LOAD	Invokes the LINK processor. This command is equivalent to the LINK and LYNX commands.
LOCATE	Determines the line number of a specified user.
LOGON	Terminates an on-line session and logs the user back on in the name and account specified.
LYNX	Invokes the LINK processor. This command is equivalent to the LINK and LOAD commands.
MAP	Invokes the LINK processor to output a load map of a run unit.
MESSAGE	Sends the specified message to the operator.

Table 7-1. IBEX Commands (cont.)

Command Mnemonic	Description
OFF	Disconnects the terminal from the system and provides an accounting summary. This command is equivalent to the BYE command.
OUTPUT	Directs object output to the specified file, or counteracts the previous DONT OUTPUT command.
PASSWORD	Assigns, changes, or deletes a log-on password for the user.
PLATEN	Sets the value of the terminal platen width and/or page length or displays the terminal platen width and page length values.
PMD	Invokes the DELTA processor to perform a postmortem dump.
PRINT	Sends accumulated symbiont output, such as output for the line printer or the card punch, to the output device.
PROCEED	Continues processing from the point of interruption. This command is equivalent to the GO and CONTINUE commands.
PROMPT	Sets a prompt character for programs that don't issue aN M$PC CALL.
QUIT	Terminates an interrupted program. This command is equivalent to the STOP and END commands.
RESET	Resets all DCBs back to their system default values.

Table 7-1. IBEX Commands (cont.)

Command Mnemonic	Description
RESTORE	Restores the previously saved main memory image. This command is equivalent to the GET command.
REW	Invokes the PCL processor to perform rewind functions.
RUN	Invokes the LINK loader and instructs it to link, load and start execution of the designated run unit.
SAVE	Saves the main memory image of the currently interrupted program on the designated file for subsequent recall and continuation.
SEND	Allows messages to be sent from the operator to the user terminal.
SET	Assigns file or device to a DCB or sets a DCB parameter.
SHOW	Displays information about the user.
START	Loads a run unit into main memory and starts execution at its beginning address.
STATUS	Displays the current accounting values for an on-line session.
STEP	Provides conditional execution of batch jobs. It operates on and tests the value of the step condition codes (SCC), which are set by the monitor or by the program performing an M$EXIT.

Table 7-1. IBEX Commands (cont.)

Command Mnemonic	Description
STOP	Terminates an interrupted program. This command is equivalent to the END and QUIT commands.
STOW	Qualifies a file to be copied to archival storage (stowed) during the system's next stow processing. The file is stowed as either active or inactive according to an option of the STOW command. If active, the original file is retained in the file system; if inactive, the original file is deleted.
SWITCH	Controls setting and resetting of the user's pseudo sense switches. With no parameters, the command displays the pseudo sense switch settings.
TABS	Sets simulated tab stops for the terminal or displays the simulated tab stop settings.
TERMINAL	Sets the terminal type for proper I/O translations.
TERMINAL STATUS	Lists the terminal type and the current values of parameters associated with its operation.
TIME	Outputs the time and date.
TITLE	Inserts a specified text on each batch page heading.
U	Causes the words UNDER DELTA to be inferred in the next command.

Table 7-1. IBEX Commands (cont.)

Command Mnemonic	Description
WHERE	Returns the line number of the specified user (if the user is logged on).
XEQ	Initiates processing of IBEX commands from a command file.

SECTION VIII

USER PROGRAM SERVICES

SYSTEM SERVICES

The services of the CP-6 monitor which are available to user programs are listed in Table 8-1. These services are available to users regardless of the language in which the user programs are written, although interface routines may be required in some languages.

System services are requested via the hardware CLIMB instruction, an instruction which protects the system from the user through both a change of protection domain and a check of all parameters passed by the user for legitimacy in his domain. With certain obvious exceptions, system services operate identically, regardless of whether the program is executing in time-sharing or batch mode.

In many cases, system services are available both via embedded program requests (as listed in this chapter) and via the job command language as detailed in Section 7.

The services are broadly grouped as follows:

- Program execution and resource control
- Memory management
- File management and logical file I/O (access methods)
- Time-sharing terminal control
- On-line diagnostic services

Table 8-1. CP-6 System Services

Name	Function
	PROGRAM EXECUTION CONTROL
M$EXIT	Exit to monitor normally; exit from command processor
M$ERR	Error current job step
M$XXX	Abort current job step
M$XCON	Give user control on exit, abort, or line hang-up
M$SEGLD	Load overlay segment
M$LINK	Call a separate program with return expected
M$LDTRC	Transfer to a program with no return possible
M$GHOST	Initiate ghost job
M$FSUSER	Find suspended user image
M$ASUSER	Associate suspended user image
M$DSI	Delete suspended user image
	RESOURCE CONTROL
M$RRES	Release resource
M$ENQ	Enqueue resource
M$DEQ	Dequeue resource
	RUN-TIME SERVICES
M$TRAP	Give user control on program traps
M$STRAP	Simulate a trap
M$STIMER	Give user control after specified time interval

Table 8-1. CP-6 System Services

Name	Function
M$TTIMER	Test interval timer
M$INT	Give user control on console interrupt
M$TRTN	Exit from trap, interrupt, timer, or exit control routine
M$MERC	Direct monitor to process any system abnormal or error code, overriding a user-specified abnormal or error exit
M$WAIT	Suspend program
M$CHECKECB	Check Event Completion Block (ECB) for completion
M$SYS	Enter master mode
M$MASTER	Enter master-protected mode
M$SLAVE	Enter slave mode
M$EXU	Execute privileged instructions
M$ALIB	Associate library
M$DLIB	Disassociate library
M$CPREF	Reset error flags
M$CPEXIT	Exit from command processor
M$ERRPRT	Output standard error message
M$SCCEXIT	Set condition codes and exit
M$SCCERR	Set condition codes and error
M$SCCXXX	Set condition codes and abort
	<u>SYSTEM INFORMATION</u>
M$TIME	Get time and date
M$DISP	Report system load parameters

Table 8-1. CP-6 System Services

Name	Function
	MEMORY MANAGEMENT
M$GDDL	Get dynamic data limits
M$GDP	Get dynamic data pages
M$FDP	Free dynamic data pages
M$GVP	Get virtual page
M$FVP	Free virtual page
M$SMPRT	Set memory protect
M$CVM	Change virtual map
M$STLPP	Get stolen page
M$RSPP	Free stolen page
M$GDS	Get segment
M$FDS	Release segment
M$SSC	Set software control flags
	FILE MAINTENANCE
M$DCB	Compile a Data Control Block (DCB)
M$OPEN	Open a file (initialize DCB)
M$OPLD	Open logical device I/O stream
M$CLLD	Close logical device I/O stream
M$JOB	Check batch job status
M$CLOSE	Close a file (terminate I/O through a DCB)
	DATA RECORD MANIPULATION
M$READ	Read a data record
M$WRITE	Write a data record

Table 8-1. CP-6 System Services

Name	Function
M$DELREC	Delete a data record or a range of data records
M$TRUNC	Truncate blocking buffer
M$CHECK	Check I/O Completion
	<u>FILE MANIPULATION</u>
M$PRECORD	Position a specified number of records forward or backward within a file
M$PFIL	Position to beginning or end of current file
M$CVOL	Close volume
M$REW	Rewind tape
M$WEOF	Write end-of-file
	<u>DEVICE-LIKE PROCEDURES</u>
M$PRINT	Write to listing log
M$KEYIN	Request operator key-in
M$TYPE	Type a message
M$LINES	Set the number of printable lines per page
M$EXTEND	Extend random file
M$CORRESPOND	Check for correspondence of DCB assignments
M$BLDCB	Allocate and build a DCB
M$DEDCB	Delete a DCB
M$SETFMA	Set file management account

Table 8-1. CP-6 System Services

Name	Function
	<u>TIME-SHARING TERMINAL FUNCTIONS</u>
M$PC	Set prompt character
M$CT	Change terminal type
M$RB	Reset break count
M$STA	Set terminal attributes
M$ACPL	Accept couple
M$RCPL	Reject couple
M$TS	Obtain terminal status
M$PURGE	Purge terminal I/O Buffers
M$CAC	Change activation character
M$COUPLE	Associate one terminal with another
M$DECOUPLE	End association of terminals
	<u>ON-LINE DIAGNOSTIC SERVICES</u>
M$DOPEN	Open DDCB
M$DCLOSE	Close DDCB
M$BLIST	Build command list
M$CIOC	Start I/O
M$DMOD#	Obtain model number and type mnemonics
M$RDERLOG	Read error log
M$WRERLOG	Write error log
M$DPART	Partition device or controller
M$DRET	Return partitioned device or controller

MONITOR ERROR MESSAGES

Each monitor error condition is associated with a unique code. This code is divided into three parts:

1) The Functional Code Group and Module ID of the area of the system that detected the error.

2) An error number uniquely identifying the error condition encountered.

3) A severity field which defines the system action to be taken if the user has not requested control of an error condition on the monitor call.

If the user has asked for control in case of errors by specifying an ALTRETURN address on his call for monitor service, then the error code is returned to him. If the user didn't ask for error control the job will either continue, the current job step will be aborted, or the user will be logged off--depending on the value of the severity field.

If the user is aborted or has initiated the logoff procedure, the monitor uses the error code to select and print an error message from an error message file. Any user who has requested error control may print the message associated with an error code returned to him. A monitor service (M$ERRPRT) is provided for this function.

SECTION IX

DEBUGGING FACILITIES

USER DEBUGGING

The DELTA debugging processor is used to debug object programs written in any source language. The language processors, in cooperation with the Link loader, supply symbolic information to DELTA such that the user can describe the debugging requirements to DELTA in terms close to the language in which the source program was 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. This allows the user to take immediate action to correct an error. In the batch mode, the user is only restricted to that which can be preplanned.

The DELTA commands may be input directly at the terminal or they may be predefined in a file.

Tracing ability is provided at several levels: on all entry points, on a specific entry point, on all non-sequential points, or on specific statements or locations. Additionally, a history mode may be set such that tracing information is saved and examined at a later time.

The command language of DELTA supports abbreviations. A user who is familiar with DELTA functions will probably find it more convenient to issue DELTA commands in their abbreviated forms. DELTA also has abbreviated command forms that match the commands of CP-V DELTA. They are not shown in the command summary below but are / for display, = for expression evaluation; TAB for indirect display, for display of previous location, and LINE FEED for display of the next location.

The general facilities provided by DELTA allow the user to:

1. Interrupt the flow of program execution on specified conditions.

2. Examine, insert, and modify such program elements as instructions and data.

3. Trace the flow of program execution.

4. Control the flow of program execution.

5. Obtain snapshot or post-mortem dumps.

DELTA's commands are classified as direct, stored, or attachable according to when they are activated. A direct command is activated immediately. A stored command is activated when specified conditions of the user program occur. Attachable commands may be issued directly or may be stored along with a stored command.

Stored commands are assigned an identification (by the user or by DELTA). This allows the programmer to dynamically alter the debugging specifications by cancelling or changing stored debug commands. Most DELTA commands allow for conditional execution.

A summary of DELTA commands is given in Tables 9-1 through 9-3.

Table 9-1. Stored Commands

Command	Function
AT	Sets a breakpoint at a specified statement.
ON ABORT	Specifies an action to be taken in the event of an abort.
ON CALL	Sets a breakpoint at the call of a specified procedure or subroutine.
ON EXIT	Specifies an action to be taken when the program exits.
ON NODE	Specifies an action to be taken when a specified overlay node is loaded.
ON TRACE	Qualifies the amount and nature of trace display information. A history buffer may be requested.
SKIP	Specifies that a range of statements is to be skipped during program execution.
WHEN	Specifies that a break in execution is to occur whenever a specified variable is modified.

Table 9-2. Attachable Commands

Command	Function
BREAK	Transfers control to the user's break key routine.
DISPLAY	Displays specified variables.
DUMP	Dumps memory.
EXIT	Exits from Delta to the monitor or a library.
GOTO	Returns to the user program.
HISTORY n	Displays the last n points from the trace history buffer.
KILL TRACE	Resets trace to the initial off conditions.
LET	Changes the value of a variable.
MODIFY	Displays and modifies memory.
NEXT	Displays and modifies the next memory location.
OUTPUT	Directs DELTA output to a specified device or file.
PREV	Displays and modifies the previous memory location.
QUIT	Exits from Delta to the monitor.
READ	Reads DELTA commands from a user-specified file.
STOP	Changes the mode of a specified stored command to stop program execution.
TRACE	Specifies the type of tracing action that is to take place.

Table 9-3. Direct Commands

Command	Function
*	Displays and modifies a memory location indirectly.
ABBREV	Defines an abbreviation for a program symbol.
DEF	Defines a symbol.
EVAL	Evaluates an expression.
FIND	Finds (and changes) a value in a specified memory range.
HISTORY n MORE	Displays n more points from the history buffer.
INSTRUCTION	Sets word addressing mode.
KILL ALL	Removes all stored commands.
KILL ATS	Removes all statement breakpoints.
KILL DEFS	Removes all user-defined symbols.
KILL INSTRUCTION	Resets addressing mode to schema.
KILL ONCALLS	Removes all stored ON CALL commands.
KILL SCHEMA	Inhibits schema usage.
KILL WHENS	Removes all data breakpoints.
M	Toggles activation mode setting.
NODE	Causes the specified overlay schema to be used.
SCHEMA	Causes DELTA to use program schemas.
SHOW	Displays stored commands and user program elements.
STEP	Steps through statements.
XEQ	Assembles and executes a GMAP instruction.
ZERO	Alters a range of memory.

Table 9-3. Direct Commands (cont.)

Command	Function
break key	Interrupts execution of an on-line terminal.
modification of stored commands	Allows the user to modify a stored command.
activation mode	Provides a highly terse form of EVAL and MODIFY group commands compatible with CP-V DELTA.

SYSTEM DEBUGGING

Extensive facilities are provided for debugging the CP-6 system itself. Facilities of specific importance are described below.

Automatic Crash Analysis

Explicit internal tests detect most software and some hardware problems and activate the system recovery routines. A dump taken at this time is submitted to a ghost job (ANLZ) which provides an analysis of the problem, including the immediate symptoms and formatted presentations of system tables and job-dependent information. About 20 specially formatted tables are presented in terms of well-known system symbols, permitting fast and accurate isolation of the problem either by home office experts or by field analysts. The analysis and repair of problems is thus especially prompt.

Remote System Analysis

The same routines used for automatic analysis may be used during system operation from any remote or local terminal to isolate system problems in a crash dump on file or in the running system itself. Further, the DELTA debugger may be used to examine and repair the running monitor, again from a remote terminal. This facility is a fast method of solving problems at CP-6 installations. It allows several home office experts to combine their various talents to solve a problem within hours. They can log onto the customer's system, gather information, analyze it, and dynamically apply the correcting patch.

Executive DELTA

An executive version of DELTA, called XDELTA, may be resident for the purpose of debugging the monitor. In addition to most of the commands listed in Tables 9-1 through 9-3, XDELTA contains special commands to reference monitor memory and to handle faults. XDELTA is independent of the monitor; that is, it performs its own I/O and does not use system services.

Symbolic Patching

XDELTA is also used to patch the monitor at boot time. The same symbolic patch format is used for both debugging and for patching. Patches are generally relocatable so that patch decks may be applied without change to all systems regardless of the configuration.

GENMD Patching

The GENMD processor permits on-line, batch, and ghost users to make permanent modifications to existing run units, thereby reducing the number of compilations required to debug a program. GENMD patches are used to modify nonresident elements of the system.

Boot-Under-The-Files

Both GENMD patches and XDELTA patches may be applied to the system at boot-time. CP-6 also provides for rereading the patch deck without disturbing the permanent files of the system and its users, which avoids the necessity of saving and restoring an extensive file system in order to apply critically needed patches.

SECTION X

TIME-SHARING

INTRODUCTION

For those activities best suited to an interactive environment, CP-6 provides time-sharing service for remotely connected terminals. A variety of language and utility processors are provided to aid the user in accomplishing:

- Program development
- Program compilation
- Program execution
- Program debugging
- File maintenance
- Text creation and editing

Programs to be executed using the other modes of operation may be completely or partially developed in the time-sharing mode. Since there is one common file management facility in CP-6, files used in the interactive mode are identical to those of the batch and real-time modes. (See Section VI for further details on the file management system.) Also, the time-sharing user has full access to spooled unit record peripherals, tape drives, and private disk packs.

The command language 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). The name of the command processor is IBEX. The IBEX language is compatible with and nearly identical to the CP-V TEL language. Users familiar with TEL will only be able to detect minor differences between TEL and IBEX.

Up to 500 time-sharing terminals may be connected simultaneously to CP-6. (This number is dependent on hardware configuration, activity at terminals, and desired response time.) The following types of terminals may be used with CP-6:

Teletype Models 33, 35, 37, 38, and 40

IBM 3270 terminals

Honeywell VIP terminals (certain models)

Any video or hard-copy terminal compatible with any of the above and with standard or APL keyboards.

CP-6 allows terminals to connect to the system via dial-up communication lines or permanent circuit lines. These lines may interface either local or remote from central site communications processors as described in the section on communications.

Regardless of how a terminal is physically connected to CP-6, terminal protocol is the same. After connection has been established, users identify themselves by entering 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, CP-6 sends an error message and requests the user to resupply "log-on" data.

An on-line session is terminated by entering a simple "log-off" command. CP-6 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.

Concepts that are relevant only to the time-sharing mode of operating are discussed in the remainder of this chapter.

SAMPLE TIME-SHARING SESSION

Figure 10-1 presents a sample time-sharing session. (The sample is a CP-V session. It reflects the general nature of a CP-6 session despite the fact that some of the details may be different.) System output is underscored and user input is not underscored. Comments describing the session are indented.

```
HONEYWELL CP-V AT YOUR SERVICE
12:00  AUG 01, '77   USER # F  LINE # 28
LOGON PLEASE: 2232,HALL
```

The system identifies itself, states the time and date, and requests that the user log on. The user types in the account number (2232) and name (HALL). To preserve security, this information is not normally echoed by the system.

```
!TEREMN_
 TERNIN
    MR\
 TERMIN
       AL STT\ATUSR\
 TERMINAL STATUS
```

The user begins to enter a command, but finds that it has been typed incorrectly; <ESC>X (which echos as an underscore and a new-line sequence) is used to delete the line. The user then begins again, makes another error, and uses the backspace mode to correct the line. Insertion of the "M" is verified by an <ESC>R which causes the line to be retyped.(R\ is echoed by the <ESC>R sequence.) As the word "STATUS" is typed, the user enters an extra "T" which is deleted with the RUBOUT key. The user then types <ESC>R to see the entire line retyped. A carriage return is pressed and the current status of the on-line terminal is displayed.

```
LINE   01
LINE SPEED 300
TIMING ALGORITHM 5
REJECT COUPLES
TRANSLATION TYPE 37
ECHOPLEX ON
TAB SIMULATION ON
UPPER CASE RESTRICT OFF
PAPER TAPE OFF
SPACE INSERTION ON
LOWER CASE SHIFT OFF
PARITY CHECK OFF
RELATIVE TABBING OFF
BACKSPACE EDIT ON
```

Figure 10-1. Sample Time-Sharing Session

```
!TABS 7
!EDIT
EDIT HERE
*BUILD INPUT
    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
    6.000  50        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
```

The user builds a file of source input, named INPUT, but notices that a line is missing. A line numbered 5.5 is inserted:

```
*IN 5.5
    5.500            GO TO 10
*END
!BUILD DATA
    1.000 1.0,2.0,3.0
    2.000 1.0,1.0,1.0
    3.000 0.0
    4.000
```

The user builds a program-data file.

```
!ANSF INPUT ON BIN
```

The user asks for a compilation of the source file INPUT with program output on file BIN.

```
ANS FORT BOO AUG 01,'77
OPTIONS>
         1:         WRITE (6,100)
        11:         END
HIGHEST ERROR SEVERITY;  O (NO ERRORS)
                    DEC          HEX
                    WORDS        WORDS
                    -----        -----
  TOTAL PROGRAM:      61         0003D
```

The summary prints at the terminal.

```
!BUILD COMFILE
    1.000 !SET F:5 /DATA;IN
    2.000 !SET F:6 UC
```

Figure 10-1. Sample Time-Sharing Session (cont.)

```
    3.000 !STATUS
    4.000 !RUN
    5.000 !STATUS
    6.000
!ECHO
!XEQ COMFILE
     0001 - !SET F:5 /DATA;IN
     0002 - !SET F:6 UC
     0003 - !STATUS
CPU = .0455 CON = 00:05:57 INT = 37 CHG = $0.25 CALS = 263
     0004 - !RUN
:P1 ASSOCIATED.
       X            Y            Z            D
     .100E 01     .200E 01     .300E 01     .374E 01
     .100E 01     .100E 01     .100E 01     .173E 01
 *STOP* 0
     0005 - !STATUS
CPU = .0715 CON = 00:06:06 INT = 37 CHG = $0.55 CALS = 412
 ***XEQ TERMINATED***
```

The user builds an execution file that, when run via the XEQ command, links, loads, and executes the program, displaying the accounting records before and after the run. The ECHO command causes the commands to be listed as they are executed.

```
!E COMFILE
EDIT HERE
*IN.5,.1
     .500 !JOB
     .600 !SET M:SI /INPUT
     .700
*RR2
    2.000 !SET F:6 UC\\/OUTFILE;OUT
*DE3
*DE5
*IN4
    4.000 !FORTRAN GO
    5.000 !LOAD (GO)
    6.000 !RUN
    7.000
*TY
     .500 !JOB
     .600 !SET M:SI /INPUT
    1.000 !SET F:5 /DATA;IN
    2.000 !SET F:6 /OUTFILE;OUT
```

Figure 10-1. Sample Time-Sharing Session (cont.)

```
    4.000 !FORTRAN GO
    5.000 !LOAD (GO)
    6.000 !RUN
```

The user alters COMFILE so that it can be run as a batch job, redirecting output to a file called OUTFILE.

```
!BATCH COMFILE
ID=00A1 SUBMITTED 12:00  AUG 1,'77
RUNNING
!JOB A1
COMPLETED
```

The user submits COMFILE to the batch stream directly from the terminal, and then checks the progress of the job.

```
!DISPLAY
USERS = 0022
ETMF  = 0001
RESPONSE 90%  < 0027 MSECS
RADS = 00440   GRANULES
```

The user requests the current system status.

```
!SEND INPUT TO INPUT.FISHER@CHICAGO
```

The user sends a copy of the file INPUT to the account FISHER, located at a remote CP-V computer site in Chicago.

```
!PCL
PCL E00 HERE
<COPY INPUT TO LP
..COPYING
<END
!M  PLEASE SEND LINE-PRINTER OUTPUT TO COPY CENTER.
SENT AT 12:00 P.M.---THANX
```

The user enters PCL (Peripheral Conversion Language) and copies INPUT to the line-printer. Returning to the command processor level, the user instructs the computer operator to send the output to another department. The operator confirms the delivery.

```
!BASIC
VER. D03
>OPEN "DATA" TO :1,INPUT
>INPUT :1,X,Y,Z
>10 D = SQR((X^2)+(Y^2)+(Z^2))
>20 PRINT " X";" Y";" Z";"    D"
```

Figure 10-1. Sample Time-Sharing Session (cont.)

```
>30 PRTNT X:Y;Z;D
>RUN
 X   Y   Z      D
 1   2   3    3.74166

>X = 1
>Y = X
>Z = Y
>RUN
 X   Y   Z      D
 1   1   1    1.73205

>SYS
```

The user enters BASIC, and issues two direct statements that access the first three values in DATA. A program is constructed and executed. The user returns to direct mode and assigns new values to the variables. The program is re-executed, illustrating the effect of the direct statements upon the program. The user then returns to the command processor level.

```
!SET F:IN-FILE DC/DATA
!AFILE.
    X            Y            Z            D
   1.00         2.00         3.00         3.74
   1.00         1.00         1.00         1.73
```

The user runs a previously compiled COBOL program called AFILE that is equivalent to the FORTRAN program above. The period after AFILE tells the command processor to fetch the program from the users account.

```
!A\

•APL
APL D02
CLEAR WS
```

The user types <ESC>A to signal the system that an APL typewheel is about to be inserted.

Figure 10-1. Sample Time-Sharing Session (cont.)

```
      )TERM 1
WAS 3

      )LOAD APLSAMPLE
 APLSAMPLE   SAVED  13:26 MAY 06,'77

      ∇INPUT[⎕]∇
    ∇ INPUT;ΔFMT
[1]   ΔFMT←14⍑0
[2]   7ρ' ';'X';11ρ' ';'Y';11ρ' ';'Z';11ρ' ';'D'
[3]   LOOP:BUFFER←⎕;⍝ READ RECORD FROM FILE
[4]   →0×⍳0=ρ,BUFFER;⍝ STOP IF END OF FILE
[5]   →LOOP×⍳2≠+/',,'=BUFFER;⍝ IGNORE ANY RECORD WITHOUT TWO COMMAS
[6]   X←∊(¯1+BUFFER⍳',')↑BUFFER
[7]   BUFFER←(BUFFER⍳',')↓BUFFER;⍝ REMOVE FIRST NUMBER AND COMMA
[8]   Y←∊(¯1+BUFFER⍳',')↑BUFFER
[9]   Z←∊BUFFER[(BUFFER⍳',')+⍳(ρBUFFER)-BUFFER⍳',']
[10]  D←((X*2)+(Y*2)+Z*2)*0.5
[11]  ,'X1,E11.3,X1,E11.3,X1,E11.3,X1,E11.3'ΔFMT(X;Y;Z;D)
[12]  →LOOP
    ∇

      )SET ⎕ IN DC/DATA

      INPUT
       X           Y           Z           D
   1.00E0      2.00E0      3.00E0      3.74E0
   1.00E0      1.00E0      1.00E0      1.73E0

      )OFF

CPU = .0532 CON= 00:18:00 INT = 68 CHG = $1.65
```

An APL program is run that is equivalent to the FORTRAN and COBOL programs above.

Figure 10-1. Sample Time-Sharing Session (cont.)

TIME-SHARING FEATURES

CP-6 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 (and the user is effectively typing blindly). Instead it is stored until the proper time for its output -- that is, the time following programatic request for input. Thus in the terminal type script, input appears following the query asking for it even if the input was typed some time previously.

Terminal Types and Timing Algorithms

CP-6 carries code conversion routines for several types of terminals. The appropriate code conversion routine is selected for certain classes of terminals during the logon sequence, but some may be changed during system operation (such as to the ASCII APL set).

Several timing algorithms are supplied which determine how many characters are to be sent to the terminal and when they are to be sent. These characters provide delay for positioning of the carrier before tabs and before and after NEW LINE characters. The number of characters is dependent on the terminal speed and the current carrier position.

Each algorithm is appropriate for a different class of terminals and may be set by the user via an IBEX command.

Transparent Mode

A program may request that no translation or other processing be performed by CP-6 on the 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, to tape cassettes, to paper tape, or to other computers.

Pagination

Output to the terminal may be formatted into standard size "pages" by the system. System service calls embedded within programs or terminal commands are used to establish page width and length. Thereafter, lines are folded to fit within the established width and a page break is created at the established length (which includes top and bottom margins and page numbering). If the terminal is a CRT, output pauses at the end of each page (screen bottom) and awaits a user signal before proceeding.

Editing Terminal Input

A series of simple keystroke commands and two special modes allow the user great flexibility in preparing terminal input. They are:

RUBOUT	Erases the last typed character.
ESC X	Erases the current input message (usually a line).
ESC R	Retypes the current input message.
ESC O	Enters the backspace edit mode. In this mode: Backspace characters move a pointer to the left on the line and CONTROL R moves the pointer to the right. RUBOUT deletes the character at the pointer and any other character replaces the character at the pointer. ESC J sets a submode in which typed characters are inserted to the left of the pointer.

Controls Over Input Conversion

Character conversions are generally done automatically by CP-6 in the proper way for each terminal type. Certain controls, however, may be set by the user. These are:

ESC U	Reverses the setting of a mode in which received lower case letters are converted to upper case.
ESC)	Causes subsequent 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

CP-6 can simulate the effect of physical tab stops on a terminal. This activity is controlled by the user with the following three independent commands:

ESC T	Reverses the setting of tab simulation mode (ON to OFF or OFF to ON). If the mode is ON, spaces are sent to properly position terminal output when a tab character is received from either the terminal or a program.
ESC C	Reverses the setting of the tab relative mode. If this mode is ON, tab stops are assumed to be relative to a left margin established by the carrier position at the last program read.
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.

In addition, the user may establish the setting of the tab simulation tab stops using either programatic or IBEX commands.

Miscellaneous Controls

There are miscellaneous terminal commands which allow the user to:

ESC F	Signal an end of file.
BREAK	Give control to the interrupt point of a program associated with the terminal.
ESC Y	Return control to the command language (IBEX).
ESC Q	Receive an acknowledgement that the system is alive.
ESC H	Temporarily halt terminal output.
ESC W	Delete all output until after the next terminal input.
ESC Z	Enter a mode in which coupled terminals may converse without their typed input being delivered to their connected programs.

(These commands and the commands mentioned previously in this section are the identical to commands offered by CP-V.)

Entry of Jobs to the Batch Job 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 may be conveniently employed. This facility allows the user to enter a job into the batch job queue for execution in the batch processing mode. The user may 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 may request the status of the job. CP-6 responds by telling the user either the number of jobs ahead of his in the queue, that the job is running, or that the job is completed. (The user may also cancel the job from the on-line terminal.) After the job has completed, the user may 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 the Computer Operator

Communication of control instructions to the CP-6 operating system is accomplished through the IBEX processor (described in Section VII). Since the on-line user is in direct control of the computing task, 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 dialogue with the individual on-line user.

For users not currently logged-on, the central operator may input a "greeting message". This message is stored by CP-6 and is presented to the user during the log-on process. Other information is placed in a "mailbox" which may be examined by the user at his convenience.

Automatic Processor Association

The time-sharing mode allows the user to work at a terminal, interacting "directly" with a CP-6 processor or with a user-written program. The word "directly" has been put in quotes because there are monitor routines which 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 variety of processors during an on-line session. There is a feature of CP-6, however, which enables the system manager to restrict a user in such a manner that interaction may take place with only one selected processor. As soon as such a user logs onto the system, the user begins interacting with that one particular processor. The 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 commands in a file and later call for their execution with a single IBEX 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 XEQ commands to any level of nesting.

Automatic Save for Line Disconnect

This feature of CP-6 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 may be continued as if it had never been interrupted.

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

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

```
PROGRAM HELD.  RECONNECT?
```

The user then responds with either Y or N. If Y, the user is reconnected to the suspended image and continues from the point of disconnect. (However, I/O going to and from the terminal may have been lost.) If the response is N, the program image continues to be retained. (The retention time is not changed

Terminal Coupling

This facility provides for coupling (linking) of two or more terminals in such a way that the input and output of each terminal is displayed on all connected terminals. All typing at these terminal keyboards (except simple input editing) appears on all of the terminals. Connection between terminals is at the message level. Usually each typed line is a message. A running program belonging to a particular terminal "sees" only the input of that terminal.

Conversations may be carried on between linked terminals by using the ESC Z sequence so as not to affect a reading program. Terminal page heading output is not coupled. The link is broken if either line is disconnected.

This facility includes mechanisms for accepting, rejecting, creating, and terminating couplings. The user may control the coupling either using IBEX commands typed at the terminal or system services embedded within the program.

Automatic Terminal Speed and Format Detection

CP-6 detects the speed and format of calling terminals by examining the first character typed by the user and sets the hardware line interface module appropriately. This eliminates the need to segregate lines by speed and results in a lower per line cost.

Terminal Tape Input

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

CTL Q or DC1 starts the reading of the tape under control of CP-6. CTL S or DC3 is sent by CP-6 to stop the tape reader.

Rubout characters are ignored during a tape read operation. This enables the user to use rubout characters to delete unwanted characters as in normal paper tape operation.

The tape read mode is set when a DC1 character is received from the terminal. It is reset (to normal processing) when a DC3 character is received.

DC1 and DC3 characters are sent to the terminal as necessary to control the input rate and ensure that buffer space is not exhausted resulting in lost characters.

A special mode is available for half duplex terminals that are reading tape. (A half duplex terminal does not respond to DC1 and DC3 characters sent by the system while it is sending tape input to the system -- even though it is connected to the system with full duplex circuitry.) While in this mode, no attempt is made by the monitor to turn the tape reader off or on. Input is accepted until available buffer space is exhausted. No program output, prompt characters, or echoes are sent to the terminal.

The half duplex tape mode is entered upon receipt of an ESC P sequence or upon receipt of a DC1 character while in the non-echoplex mode. The mode is exited by a balancing ESC P sequence or by a DC3 character if it was initiated by DC1.

SECTION XI

MULTIPROGRAMMED BATCH PROCESSING

INTRODUCTION

CP-6 offers a comprehensive multiprogramming batch facility for those jobs which do not need or do not benefit by on-line processing (time-sharing or transaction processing). Multiprogramming allows more than one batch job to be run at the same time, distributing the resources among the jobs. Resources include CPU time, memory space, tape drives, and private disk spindles. Several batch jobs may be run concurrently under the control of the Multi-Batch Scheduler.

In CP-6, as in other operating systems, concurrent operation of several jobs (multiprogramming) 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 can be overlapped providing they are directed to separate devices (which is often the case). This overlap applies not only within the batch mode but also with all other jobs. Thus maximum use of CPU, disks, tapes, and other system resources is achieved.

SOURCES OF BATCH JOBS

Jobs to be run in batch mode arrive from three sources in CP-6: local card readers, remote batch stations somewhere in the communication network, and the users of time-sharing terminals. As described in the remote batch chapter, CP-6 itself may act as a remote batch station; thus one CP-6 system may submit jobs to another. Jobs may also be received from IBM systems through 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 JOB SCHEDULING

Batch jobs are protected from execution-time resource conflicts or deadly-embraces by the CP-6 job scheduling algorithms. Each batch job contains a command which specifies the required resources and the system ensures availability 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.

CP-6 can concurrently run a large number of batch jobs. Each job is run in a 'batch stream' (see glossary). The number of such streams is controlled by the system manager who usually finds 32 streams sufficient to provide full resource use and flexibility to adapt to other circumstances such as evening, night, and weekend shifts.

Each of the batch streams has a defined resource profile which is dynamically changeable by the system manager. This profile describes the ranges of job resources which are allowed when running in each stream. Thus in order to qualify for a given stream, the resource requirements of a job must match the stream's profile. It is possible for a job to "fit" in many streams. Whenever a stream is empty (not running a job), the scheduler selects the first (highest priority) job from the batch input queue whose requirements fit the stream's profile. Both the job and its required resources are assigned to the stream and the job begins execution.

FACILITIES FOR BATCH JOBS

Batch jobs, like all other CP-6 jobs, make use of standard system facilities. These include:

1. File management described in Section 6.
2. Common programatic services described in Section 8.
3. A common command processor described in Section 7.
4. Debugging facilities described in Section 9.
5. Processors described in Section 3.

Cataloged Procedures

Especially important among the facilities provided by the system and its command processor is the ability to create a file of job 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 (referred to as a symbionts in CP-V) is provided to help eliminate bottlenecks associated with slow speed peripheral devices. Symbionts are 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 provides for complete buffering between I/O devices and the user's program. Therefore, a user's program never has to wait for an I/O device to complete an action. 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 11-1.

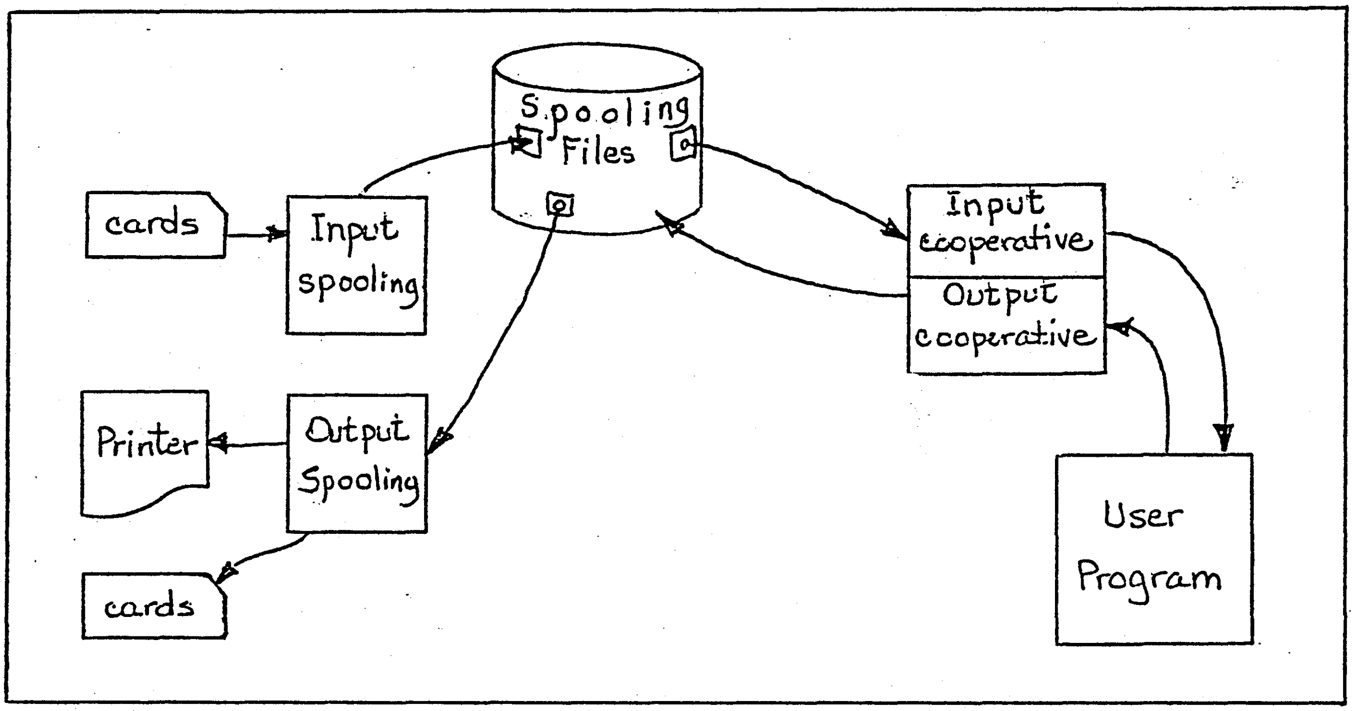

Figure 11-1. CP-6 Spooling

Spooling files are normally written to disk completely 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-hour job which produced 14 hours of printing uniformly over the execution time of the job will complete in about 14 hours rather than 28 hours.

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.

Spooling operations are controlled by the central site operator (or, to a certain extent, by remote batch terminal operators) using commands described in Section 19.

LOGICAL DEVICE STREAMS

A logical device stream 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 device streams may be defined by the user by means of the LDEV command. There are three logical device streams available for use without specific user definition. They are used for control command input, line printer output, and card punch output.

Logical device streams have three major uses. 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 (device name, form name for printer, number of copies, etc.). Third, the three predefined streams allow users to run jobs with little concern for the physical location or connection of devices on the system.

SECTION XII

REMOTE PROCESSING

INTRODUCTION

The purpose of CP-6 remote batch processing is the management of jobs and data submitted from remote sites. Jobs are sent from a remote site to the central site, are processed there, and may direct output to the originating remote site, a central site device, or another remote site as specified. Remote terminals can range from a simple card reader and line printer combination to another large-scale computer system with an assortment of peripheral devices. Important features of the remote processing system include:

- Any user of the CP-6 system can communicate with any number of devices at one or several remote sites. When data is being sent by a user program to a remote site, the remote site need not be connected since CP-6 automatically buffers on disk for deferred transmission.

- Through monitor and user interfaces, virtually any type of device (e.g., tape, disk, plotter) may be accessed at a remote terminal.

- A remote site may be another large-scale computer, and files of data may be transferred between user programs at the central and remote computers.

- A CP-6 system may act as the central site to some remote terminals and as a remote terminal to other computers, simultaneously. Thus it may be a master of some terminals while being a slave to several computers. This feature encourages the construction of communications networks.

- "Remote stations" need not be remote nor are they necessarily composed of physically associated devices. Local peripherals may be grouped under the workstation definition mechanism - associating, for example, a particular printer and a particular card reader into a designated "remote station". Further, a peripheral from one physical site may be associated with a peripheral from another physical site to form a single "remote station".

- Definitions of workstations (remote stations) can be added, deleted, or modified during system operation.

REMOTE PROCESSING TERMINALS

Two basic types of remote terminals are supported by CP-6: Remote Batch Terminals (RBTs) and Intelligent Remote Terminals (IRBTs).

An RBT is typically a card reader, card punch, and line printer combination which is used primarily to allow batch processing I/O functions to be performed at remote sites. That is, a job is input to the system from the remote site card reader, the job is processed at the central site, and the output is sent to the remote site line printer or card punch. The output may optionally be directed to the central site or to another remote site.

An IRBT can be either a mini-computer system for which the primary function is to control the operation of peripheral equipment (e.g., COPE 1200) or another large-scale computer system (e.g., another CP-6 system). Any computer system that supports the IBM HASP Multileaving protocol may act as an IRBT to CP-6. (This and all other reference to "HASP" and "Multileaving" in this document refer to the HASP Multileaving protocol as described in HASP Version 2.3 program documentation and not to the IBM HASP operating system, except where IBM HASP is specifically noted.) Multileaving allows a single data block to contain records associated with different peripherals at the IRBT. In conjunction with a feedback mechanism that temporarily suspends transmission for a single peripheral, multileaving permits peripherals of different speeds to operate at their individual rates.

The majority of records transmitted to and from any computing system contain many adjacent characters that are identical. CP-6 and supported IRBT systems increase throughput by contracting (compressing) strings of characters before transmission and expanding (decompressing) these character strings after receipt. Thus, while the compression/decompression process improves overall system efficiency, the user of transmitted data does not have to be aware of its source or format.

Hardware Connection of Remote Terminals

A remote terminal is connected to the central site over a communication line that is either a private line or a switched line.

If the connection is long enough, two modems are required. The modems provide interfaces between the line and the remote terminal and between the line and the central site.

As described in network processing chapter, batch terminals for remote processing may be connected to a local communications processor (CP) of the CP-6 host or may be connected to a remote communications processor (RCP) in the CP-6 network. Peripheral devices local to the CP-6 host or to one of the CPs or RCPs may be incorporated into remote processing workstations along with RBTs and IRBTs.

Figure 12-1 depicts the relationship between the central site, remote sites, CPs, RCPs, and the peripherals of each. Not shown in the figure, but available as discussed in the network processing chapter, are time-sharing and transaction processing terminals on the CPs and RCPs.

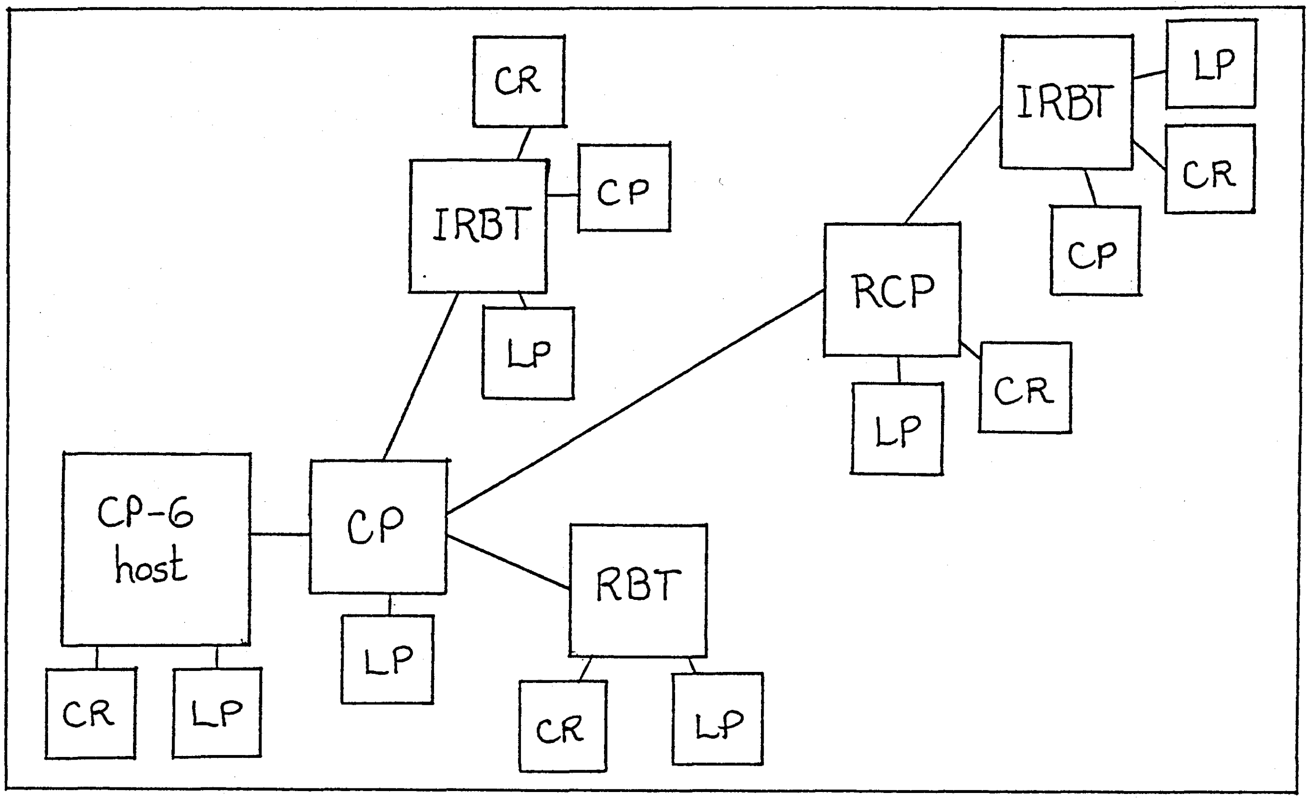

Figure 12-1. CP-6 Remote Processing.

REMOTE PROCESSING MODES

The remote processing system is designed so that the CP-6 system may act as the central site to a set of remote terminals while simultaneously acting as a remote terminal to one or more other systems. A system that is acting as the central site is referred to as the "master" system, and a system that is acting as a remote terminal is referred to as a "slave" system.

To the CP-6 system, the role of master and slave manifests itself only at log-on time. After the modem to modem communication path is established, the slave logs onto the master. (The master cannot log onto the slave.) Once the log-on is complete, the communication path between the master and slave is symmetrical - streams of data flowing in both directions over the communication path.

Four fundamental modes of remote processing are

1. A CP-6 master system connected to one or more slave RBTs.
2. A CP-6 master system connected to one or more slave mini-computer IRBTs.
3. A CP-6 system communicating with another CP-6 system.
4. A CP-6 system acting as a slave IRBT connected to another computer system acting as the master computer. (These are primarily IBM HASP systems, but also may be more modern network systems.)

These four modes may be combined to provide a large variety of communications networks. An example of such a network is given in Figure 12-2. (The arrows point to the RBTs and slave IRBTs.)

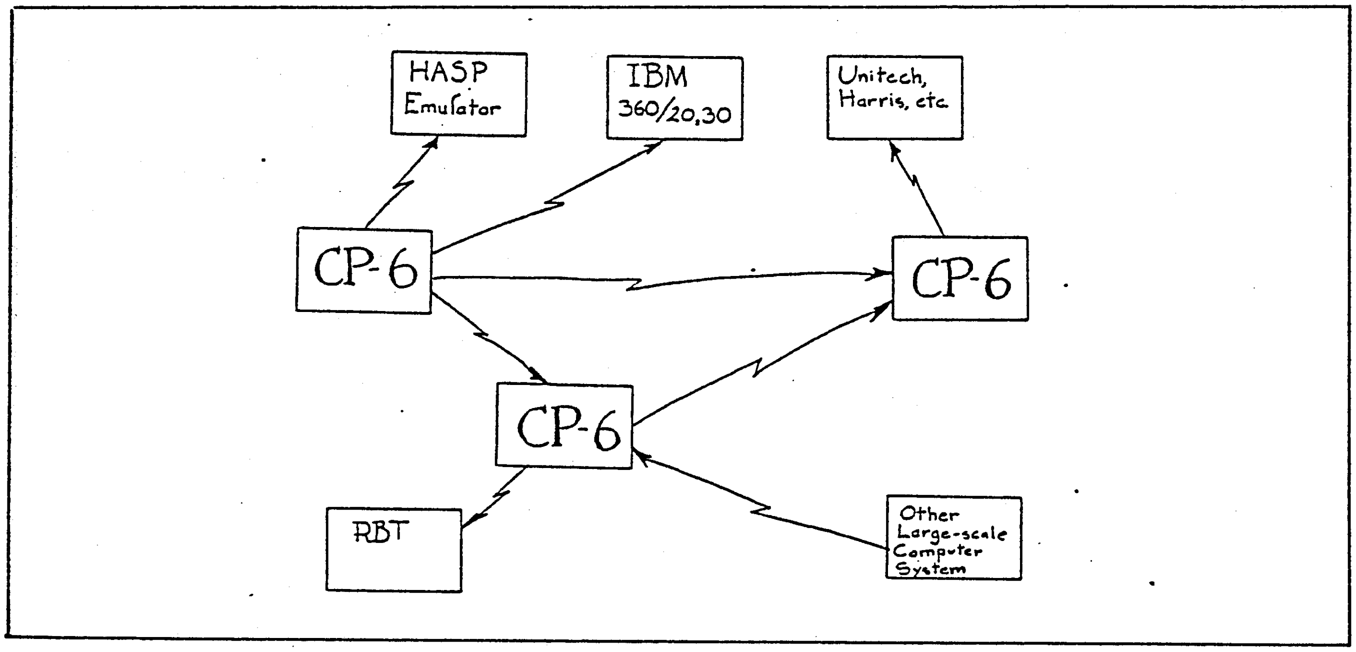

Figure 12-2. CP-6 Remote Processing Network.

WORKSTATIONS

The CP-6 system contains a description of the hardware characteristics of all remote and local devices. A workstation is a definition which groups the devices into logical collections that are independent of physical association. (A particular device may be part of more than one workstation definition.) A workstation, then, is a named data structure contained in a CP-6 file which represents a group of hardware characteristics at local or remote sites. (In sophisticated applications, a workstation definition may include pseudo-hardware.) The workstation may represent all of the hardware at a given remote site, the hardware available to a particular group of users at a given remote site, or the hardware available to a particular group of users at any site which has the requisite configuration. Further, a workstation may logically group together devices from several physical terminals and may include peripheral devices local to the L66 CP-6 hosts or L6 CP-6 communications processors. The definition of a workstation specifies such items as:

- Name of the workstation.
- Maximum priority for jobs submitted from the workstation.
- Whether the workstation will be a slave computer or the master computer.
- Type of teminal(s) to be used.
- Peripheral devices to be associated as part of the workstation.
- Attributes of devices defined for the workstation.

Each workstation is given a workstation name (WSN) of one to eight alphanumeric characters. Local devices at a CP-6 system have the workstation name LOCAL.

A workstation definition may be created dynamically any time during system operation by the system manager. The attributes of workstations may also be changed or workstations may be deleted during system operation.

Operator Control Commands

There are several control commands that are available only for remote processing. These commands are used by the remote site operator to gather information about and control his terminal. Table 12-1 lists the commands that exist in CP-V. A similar set of commands will exist for CP-6.

Table 12-1. Remote Processing Monitor Control Commands

Command	Function
RBID	Logs a workstation onto the system.
RBDISC	Logs a workstation off the system.
RBXXX	Logs a workstation off the system and disconnects it immediately.
RBMSG	Transmits a message to the central site operator.
RBDEV	Displays the status of all devices at the workstation and, where applicable, the name of the form mounted on each device.
RBINFO	Displays various CP-6 statistics.
RBPRIO	Changes the priority of the workstation's files in the symbiont system to a specified priority.
RBHOLD	Prevents current output files and output files from other sources from being output, but does not affect input files (except that results from the execution of such files is held).
RBRETRIEVE	Releases files that were held with RBHOLD.
RBDELETE	Deletes input, output, or executing files from the symbiont system.
RBSTATUS	Requests the status of files belonging to the workstation.

Table 12-1. Remote Processing Monitor Control Commands (cont.)

Command	Function
RBSWITCH	Changes the workstation assignment of output files.
RBSUSPEND	Suspends output on specified device(s).
RBCONTINUE	Specifies that suspended output is to be continued from where it stopped.
RBREPRINT	Restarts output at the symbiont retry point.

Transferring Files to Remote CP-6 Systems

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 PCL commands (which replace CP-V ISCL commands), the user must be authorized at the remote site as well as the local site. User requests may

- Copy a file from a remote CP-6 system.
- Copy a file at a remote CP-6 system.
- Delete a file at a remote CP-6 system.
- List file(s) which are at a remote CP-6 system.

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

These commands may be grouped together in a file and "standing orders" may cause the filed commands to be executed at predetermined times. Thus files may be transferred between systems on a regularly scheduled basis.

SECTION XIII

NETWORK PROCESSING

INTRODUCTION

CP-6 communications facilities are a significant extension over those of CP-V. This is due to the distribution of communications processing into a network of Level 6 minicomputers together with expanded software architecture which takes advantage of this distribution. Communications networks may be geographically distributed and connected to several CP-6 host processors as well as to a variety of terminals. Figure 13-1 shows a sample CP-6 communication network.

CPs are local communications processors physically connected close to the CP-6 host. RCPs are remote communications processors connected to local CPs via high-speed communication lines.

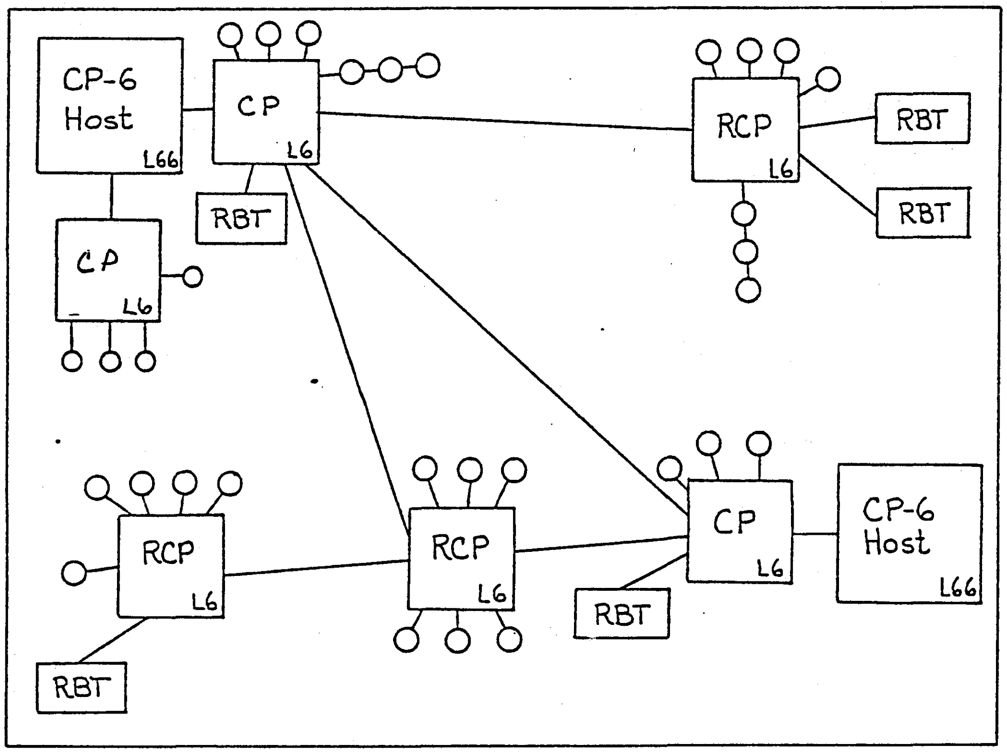

Figure 13-1. Sample CP-6 Communication Network

The objectives of CP-6 communications are:

1. To geographically distribute the communication processing load.

2. To provide fast interactive response to each terminal directly from the CP or RCP which is physically closest.

3. To provide effective error detection and recovery over the long high-speed lines between CPs and RCPs.

4. To provide device independence in the CP-V style by converting the I/O to each of the connected terminals into a common network protocol so that programs need not know the characteristics of the terminal to which they are connected.

5. To provide for communications between CP-6 hosts for file exchange.

6. To provide cost effective communications by concentrating low volume traffic into efficient high bandwidth trunks.

CP-6 communications processing relates directly to other subjects discussed in this manual.

1. Timesharing. The communications network is responsible for providing the terminal interface and thus the personality of CP-6 terminals. Time-sharing is discussed in Section 10.

2. Remote Batch Processing. Similarly, the interface programs which allow connection of remote processing terminals are programs in the CP-6 L6 network. Remote batch processing is discussed in Section 12.

3. Real-Time. A major portion of the real-time services of CP-6 is distributed on Level 6 minicomputers. These may be local to or remote from the host L66 CP-6 system. In either case communications with other real-time programs, whether in L6 or L66, is carried out using CP-6 communications network services. Furthermore the base system software is common for both real-time and communications applications residing on the L6 computer. Real-time processing is described in Section 15.

4. Transaction Processing. Terminals connected into groups by the communications system may be controlled by transaction processing programs in a host. This method of connection is such that terminals may move freely from one transaction processing application to another or to time-sharing use as required. Transaction processing is described in Section 14.

5. Operator Consoles. CP-6 operations can be controlled by use of a group of terminals, each assigned to a particular function (tape mounting, printer control, user control, etc.). These terminals are part of the communications network and may be anywhere on that net. Computer operations are described in Section 19.

The remaining sections of this chapter discuss how CP-6 networks are configured, how terminals and programs are connected to each other, how programs talk to terminals (the protocols), and the facilities for reliability and recoverability.

NETWORK CONFIGURATIONS

As stated above, CP-6 networks are composed of Level 6 computers. There are two basic types of local and remote communications processors: CPs and RCPs. 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 Level 6 memory and Level 66 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 also may be used to increase reliability of connection. CP-6 uses a full-duplex bit-oriented protocol over these lines for maximum throughput and control over errors. The protocol used is Honeywell's HDLC which is similar to IBM's SDLC.

Whether local or remote, each communications processor carries the software required to interface all supported types of terminals to the network. As detailed in other chapters, these include asyncronous terminals such as Teletypes and typewriters, multidrop message oriented terminals such as CRTs, remote batch terminals, and HASP-protocol terminals. It is the task of this interface software to transform the characteristics of each of the supported terminals into the standard protocol of the network.

Each terminal which connects to the network logs on following connection. This process dynamically establishes the proper parameters for the terminal. Logon information is created dynamically during system operation and thus may be changed without any sort of system definition (SYSGEN) process. The network configuration and capabilities of CP-6 are adjusted and augmented during system operation rather than by an off-line reconfiguration process.

The network may provide connection to more than one host L66 computer. Terminals are connected to a particular host during the logon process, either automatically through specifications in a user's logon profile or at the user's explicit request.

The network is logically independent of the hosts that connect to it but it is physically dependent on one or more hosts for initialization, logon files, and many other services. Once initialized, however, the network can continue to function during times when one or all of its hosts are out of service, picking up where it left off (within certain limits) when the host returns.

As in CP-V remote processing, CP-6 hosts can connect together for purposes of remote job entry, job load leveling, or file transfer. CP-6 networks may connect to other networks or systems such as CP-V, HDNA, SNA, and Telenet. The extent, timing and priority of these connections is yet to be determined.

CONNECTING TERMINALS TO PROGRAMS

In CP-6 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 multidrop line. CP-6 communications processing controls polling on such lines, not the user program. This isolates applications from device-specific characteristics and makes it possible, for example, to use one of several 3270s 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. Unlike CP-V, a CP-6 workstation is not a description of a physical device, but rather a logical collection of physical devices. This kind of workstation extends to the local peripherals of the L66 such 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.

Asyncronous terminals, regardless of speed and format, may connect to any asynchronous CP-6 line. An automatic speed and format detection procedure determines the terminal speed (from 100 to 9600 baud) and sets up the line interface unit appropriately. This Autobaud technique reduces per-line cost and administrative overhead in dealing with groups of fixed speed lines.

Terminals logon to CP-6 in two stages:

1) Network Logon
2) Host Logon

However, the fact that the two stages happen may not be visible to the user.

Network Logon is controlled by the contents of the 'SUPER' file which is maintained by the Network Manager. When a terminal connects to the network, the logon identification is collected and supplied to the Network Logon process over a previously defined path. Information returned from Network Logon refines the characteristics of the caller using previously stored information, such as the exact device complement of a particular RBT and to which host connection should be made. Connections are now set up between each terminal device and the appropriate host.

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 Network Manager via modifications to the contents of his 'SUPER' file. No 'SYSGEN' or initialization process is required except that which establishes the path to Network Logon. Shifting activity from one host to another is simply a matter of modifications to the 'SUPER' file.

After the proper host is determined, Host Logon is consulted to determine how each connected device is to be used. Host Logon uses a 'SUPER' file to determine what the system use is to be for each device. This file, controlled by the System Manager, causes the device to be treated in one of two ways:

1. If the device is to be considered a master (time-sharing) terminal, a new user is started and the ID (or yet another translation obtained from the file) is passed to the control program which accesses the :USERS file and performs the accounting logon which includes accounting initialization. Thus a time-sharing user actually goes through three "logons":

   a. The Network Logon which determines that he is at a Teletype and to which host he is to be connected.

   b. The Host Logon which determines that he is a time-sharing user.

   c. CP-6 logon in the host L66 which fills in his JIT from the :USERS file.

2. Any device which does not represent a time-sharing user can be referred to by the general term "slave line". This includes:

   a. Operators' consoles

   b. RBT (symbiont) devices

   c. Transaction processing acquired stations

   d. Remote peripherals (other devices, accessed by programs like local peripherals)

   Operators' consoles may need to be connected to KEYIN; RBT devices to the input or output symbiont; transaction processing terminals to an acquisition pool. All of this hookup is performed by Host Logon following instructions in its control file.

This connection process, described for connecting terminals to programs, is also used to connect programs in one host to programs in another host. Both programs may be L66 CP-6 hosts, both may be L6 CP-6 real-time hosts, or one may be a L6 and one a L66. Thus the needs of real-time programs for communication with each other are served by the same facilities which service terminal communication.

VIRTUAL TERMINAL ACCESS METHODS AND COMMUNICATIONS PROTOCOLS

In order to provide extensions in CP-6 so that programs may take advantage of the rich features of modern CRT terminals and at the same time continue the device independence and familiar READ/WRITE method of talking to devices or files used in CP-V, a set of virtual protocols or access methods will be introduced in CP-6. 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 in such a way that a user can either write programs which work correctly regardless of the access method or end device or he may use an access method which takes full advantage of the capabilities of a particular terminal class. Not all uses of the most complex access method will be implemented "perfectly" on every device, but some action will take place. For example a completely satisfactory mapping of a "cursor home" function is not likely on a Teletype. Access methods to be available in CP-6 are:

1. Teletype or Unit Record. The device (as in CP-V) looks like a Teletype or printer/card reader combination. Vertical format control, tabs, etc. apply.

2. CRT. A new access method allowing use of modern CRT terminal features such as cursor movement, high light, blink, etc.

3. FORMS. A generalization of CP-V TFD formatted terminals. (See the transaction processing section.)

4. INTER-USER. Features for conversation between user programs.

5. TRANSPARENT. Direct access of an actual physical device.

It is important to note that in choosing an access method the user does not change the operations used to converse. They are still READ, WRITE, OPEN, and CLOSE. Exactly what they do varies of course, just as it does with the access to keyed or consecutive files, but it is always possible to program in such a way that the variations are of no concern. Additional operations are available in some access methods to control special features of terminals, just as the set prompt (M:PC) and set terminal attributes (M:STA) were used in CP-V. These however are 'device independent' in that the system extracts meaning as appropriate for a particular device even if the meaning is simply ignoring the command.

So that all access methods be convertible to the needs of all terminals, a common protocol is used in CP-6. Each access method is interfaced to the common protocol. At the destination terminal, the common protocol is transformed into the needs of the specific terminal. This provides a standard way to provide interfaces for new terminals as they are developed or to connect 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, CP-6 provides private networks of terminals called Communication Groups (CG). CP-6 communication groups have the following properties:

- A single DCB may connect a program to many devices or terminals in the CG.

- Many separate programs (CP-6 jobs) may connect to a CG and each may have an address on the group.

- There is a special read operation which delivers the next message (arriving from any of the terminals in the group) to the reading program.

- The group may be composed of devices or terminals from anywhere in the CP-6 network, unrestricted by physical characteristics.

- Terminals may dynamically join and leave the group.

- There is, optionally, a file-backed queue of messages associated with each CG in which messages awaiting processing may be stored.

Communication groups are used in standard CP-6 processing for input symbiont terminals, output symbiont terminals, and operator console terminals.

RELIABILITY, RECOVERABILITY, RECONFIGURATION

As mentioned previously, the CP-6 network is independent of its hosts. This is not completely true since the network is indeed initialized from some host and the network depends on hosts for services which require file services such as Network Logon. But the network is able to sustain itself when a host crashes and ride through the period, minimizing the effect on connected users.

The network's ability to make a host crash unnoticeable at terminals depends on the terminal type and the system options selected.

- Terminals connected to one host will not be affected if another host crashes.

- Transaction processing terminals may use a journalizing service in the network to store transactions while the host is recovering. In this case, terminal users will not be aware of the down host except possibly through slow responses.

- Crash of a single machine in the network is automatically recovered. Users of other parts of the network are unaffected.

- Lines between nodes in the network may be added, deleted, recalled, etc. without loss of data when reestablishing reliable communications.

Diagnostics and dynamic verification programs are available for use on the machines in a CP-6 network. Diagnostics may be used during system operation to isolate faults in parts of the system to the extent that the machine is "well enough" to execute diagnostics. Verification procedures are run periodically to check and report on the status of communication lines, thus making bad lines visible to customer engineers. Errors, especially correctible errors, are reported to error logs where they form a profile useful in predicting trouble areas or lines.

Because of the dynamic terminal identification processes mentioned above, almost all additions of terminal devices to the system are done during system operation. No network shutdown is necessary to add capability for an additional terminal. Also, because the Level 6 software is capable of adding programs dynamically by down-time loading them from a host, the system may add handlers to accommodate a new terminal type without a network shutdown.

CONFIGURATION AND BANDWIDTH

Two factors limit the number of terminals in a CP-6 network: physical connectability and processing bandwidth. Physical connectability, both hardware and software limits, are much easier to state accurately than are bandwidth limitations which depend on hardware and software speed as well as on the nature and amount of data which arrives from each terminal in each time period. CP-6 has, for example, a much higher capacity when processing print lines destined for a remote printer than it does when processing input characters from a time-sharing terminal -- by an order of magnitude or so.

CP-6 networks have physical connectability for tens or hundreds of L6 nodes and for thousands or tens of thousands of terminals. Software is limited by similarly large numbers.

Processing bandwidth is load dependent as stated above. A working rule of thumb is: a processing bandwidth of 15K char/sec for Level 6 hardware and character mixes typical of CP-V time-sharing and remote processing operation. This translates roughly to 128 time-sharing lines and four RBT terminals per Level 6 communication processor. Both much higher and much lower capacity may be expected in the reasonable range of CP-6 applications.

SECTION XIV

TRANSACTION PROCESSING

INTRODUCTION

Transaction processing (TP) is a unique combination of hardware and software designed to satisfy the data processing requirements of business and the professions. It is an optional feature of the CP-6 system. Transaction processing combines the hardware field of teleprocessing with the software fields of operating systems, data management systems, and special transaction processing components. Critical to the entire operation is network processing which allows information to be transmitted over communication lines between human and computer, computer and computer, and human and human.

In the software realm, transaction processing requires the standard operating system functions of job queuing, scheduling, and execution. It also requires the current sophisticated data management systems that have evolved from the basic storage and retrieval of information. This generalized software is further enhanced with transaction processing components that interface with user stations and with software designed at an installation. The result of this hardware and software combination allows business and the professions to query a central data base or to dynamically modify its contents as transactions are submitted to the system throughout the organization.

Transaction processing consists of the set of CP-6 components shown in Figure 14-1. Integrated Data Storage (I-D-S/II) is part of the CP-6 system but its use in transaction processing is optional.

All inputs to TP are messages, and all output are reports or acknowledgments. The entry of a single message may produce an acknowledgment, a single report, or several reports, depending on the user-supplied software module that processes the input. The entire processing cycle is called a transaction.

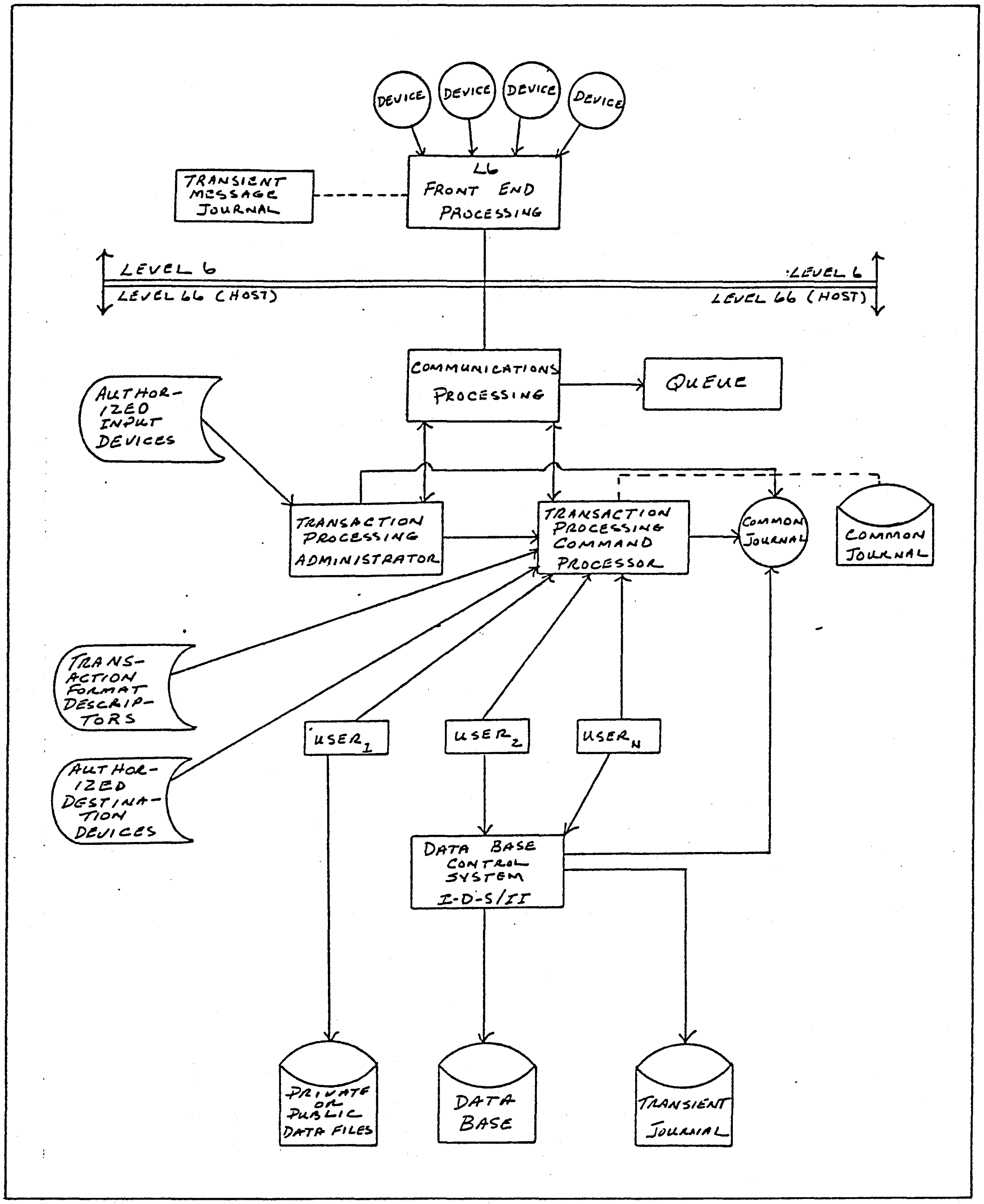
DEVICE
DEVICE
DEVICE
DEVICE
TRANSIENT MESSAGE JOURNAL
L6 FRONT END PROCESSING
LEVEL 6
LEVEL 66 (HOST)
LEVEL 6
LEVEL 66 (HOST)
COMMUNICATIONS PROCESSING
QUEUE
AUTHORIZED INPUT DEVICES
TRANSACTION PROCESSING ADMINISTRATOR
TRANSACTION PROCESSING COMMAND PROCESSOR
COMMON JOURNAL
COMMON JOURNAL
TRANSACTION FORMAT DESCRIPTORS
AUTHORIZED DESTINATION DEVICES
USER1
USER2
USERN
DATA BASE CONTROL SYSTEM I-D-S/II
PRIVATE OR PUBLIC DATA FILES
DATA BASE
TRANSIENT JOURNAL

Figure 14-1. Transaction Processing Components

FRONT-END PROCESSING

Front-end processing is an L6 system component. It controls the communications network within its domain, and provides special transaction services for transaction processing users. Once TP has been initialized, the front-end processor acts as a complete system interface for the TP stations in the communications network. For example, it logs user stations on and off, accepts messages and edits them through Transaction Format Descriptors (TFDs), and journalizes messages. Thus, the front-end processor accepts transactions in a multiplexed fashion from the stations it controls and routes the resulting reports back to the originating stations or to alternate destinations.

STATION NAMES AND REPORT DELIVERY FILES

The TP Administrator, a module supplied by Honeywell, reads the Authorized Input Device File to identify stations that it may control during the course of transaction processing. This file associates logical names with logical stations, specifies any required passwords, and may restrict the types of transactions entered from certain stations. A logical name is one entered by a station user as he logs onto the system. A logical name may also identify a station to which a report is to be delivered, even though the transaction that produced the report did not originate at that station. The Authorized Input Device File is created by a utility processor.

A report may be sent back to the station that originated the transaction, or a report may be routed to the location of another station or a group of stations in the communications network. Whenever a report is to be delivered "to location", the TP command processor module selects the appropriate TFD and submits it, through the communications path, to the front-end together with the body of the report. It also transmits destination information, obtained from the Authorized Destination File, to the Communications Processor.

The TFD File

The input/output Transaction Format Descriptor file (TFD) consists of records which describe the form layout of each TP message and report. At least one TFD must be prepared for each message type and each report type in the system. For messages, TFDs are used to prompt the user for specific information and to detect errors typically made by a station user, such as entering data in the wrong field or omitting required information. For reports, TFDs control the report layout (titles, columns, rows) so that applications programmers are not required to format reports within the application program. The TFD processor, which generates this file, permits TFDs to be inserted, modified, or deleted as required by the application program.

OPERATOR CONTROL

If TP is included as a component of the operating system, it may be initiated or terminated at any time by the installation manager, or it may be invoked automatically at a specific time of day depending on System Definition parameters. It will be automatically re-invoked at system re-start following interruption of operation. Once the system is operational, TP may be serviced or controlled from both the operator's console and the master terminal via a set of commands specific to each of these terminals.

COMMON JOURNAL

Messages and reports may be recorded in the common journal for purposes of maintaining audit trails, for data collection activities, or for recovery in the event of a system malfunction. Use of this journal is optional. When used, it provides a complete history of the flow of transactions and reports through the system, via the date and time stamped on each record. If the installation elects to journalize transactions or reports (or both), then in the event of a malfunction, any affected transactions and reports may be reprocessed, as required, from the common journal.

TRANSIENT JOURNALS

Transient journals are optionally maintained at each link between hardware processors, such as at the link between a front-end processor and the host. This provides another level of data integrity control for those users who require it.

SYSTEM QUEUE MANAGEMENT

System queue management resides within the communications processing area. Initially it receives a list of transactions together with their priorities, from the TP Administrator. These priorities will dictate when messages and reports are to be processed and, coincidentally, which application modules are to be scheduled for the user slots. System queue management enters and removes entries from the queue, but the TP Command Processor has control privileges which permit it to request the insertion or deletion of any entries associated with transaction processing activities within its scope.

When reports and messages have been completely processed, they are automatically deleted from the queue. This technique frees the queue space and reduces the probability of "losing" messages or reports within the system. In typical TP usage, all transactions and reports are automatically queued.

System queue management is always part of the system, and thus queuing may be used for purposes other than transaction processing.

TRANSACTION PROCESSING ADMINISTRATOR

The TP Administrator is a privileged ghost processor (see Glossary). During system initialization or re-start, it performs certain one-time tasks such as opening the TP system files and initializing system tables. Thereafter, the TP Administrator checks the log-on ID of each TP terminal, verifies passwords, determines which TFDs are pertinent to each and when appropriate passes this information back to the Front End Manager. The TP Administrator also keeps continuous statistics, determines when to send messages to the master terminal or to the operator's console, and responds to commands from these devices. As needed, it calls the appropriate system components to perform queuing, journaling, scheduling, processing and recovery. It also controls the order of message processing, based on priority, and makes the required adjustment whenever it is notified of a dynamic change in the priority of a message.

TRANSACTION PROCESSING COMMAND PROCESSOR

The Transaction Processing Command Processor (TPCP) is the Honeywell-supplied interface to TP applications programs at the installation. The actual processing of transactions and the creation of reports occur in user-written modules (which may be written in any CP-6 supported language). The TPCP is a shared processor consisting of routines that transfer messages and reports to and from the user-coded programs and the queue management system. In addition, the TPCP maintains user exit-control so that it is able to intercept any user errors and take the appropriate action.

I-D-S/II

Integrated Data Store/II (I-D-S/II) is a data base management system that allows an integrated data base to be accessed by many independent users, providing each user with a separate subset view of the total data base structure. The full logical structure of the data base is described by means of a schema Data Description Language (DDL). The physical characteristics are expressed in a schema Device Media Control Language (DMCL).

For I-D-S/II, the Data Base Administrator (DBA) is a person or group of individuals who control the design, creation, access controls, and the use of the schema file and the data base files. Once the data base design has been successfully translated by the appropriate functions of the Data Base Administrator Control System (DBACS), then subschema views can be prepared. The DBA has the responsibility for determining the scope of access to any subschemas created for a data base.

The Data Manipulation Language (DML) facility of I-D-S/II provides a means by which the user can access a data base that has been described by a schema. The facility includes special registers, status indicators, conditions, record selection expressions, currency indicators, subschema selection expressions, exception handling mechanism, privacy protection mechanism, and procedural statements. The DML relies on a host language, frequently COBOL, to manipulate data before storing it or after retrieval from the data base.

After a schema has been translated, device characteristics added, a subschema translated and validated, data base files allocated and initialized, and a program compiled using the subschema, then the data base can be accessed to store and maintain data. A validated subschema provides enough information to the compiler to allow the successful compilation of permitted DML statements, using the subset of the data structure and contents described in the particular subschema. The program deals with those fields and uses the formats specified in the subschema. Any alias names declared in the subschema become the only valid names available to the program.

The program can be made independent of many changes to the schema by the use of data descriptions tailored to the requirements of the program. Any transformations needed to convert the fields from the format of the data base (schema view) into the format used by the program (subschema view) are established in the validation process. If the data base formats change, then only the subschema needs to be revalidated and the resultant control structure must be relinked with the program object deck. The program need not be recompiled.

SYSTEM INTEGRITY

The integrity of the central data base, and therefore of the entire system, depends on the inclusion of specific records in the common journal. When journalization of certain records is specified, other records are automatically journalized as diagrammed in Figure 14-2. (All common journal records are listed and defined in Table 14-1.)

If an installation elects to journalize transactions, reports, or both, then transactions and reports being processed when a failure occurs may be reprocessed from the common journal. Should a failure occur, the data base administrator must decide which of the recovery phases to execute and the central operator initiates the recovery process.

RECOVERY

Recovery is an important aspect of transaction processing since the system must be capable of detecting failures and recovering from them. Some failures that can occur are operator errors, machine and device failures, accidental destruction of the data base of the transaction queue, file contention problems, and violation of rules established for system usage. Errors may also occur in user modules written at the installation.

Recovery handles several types of failures. It handles the abort of a single transaction, the abort of a system component, and a crash of the system itself. Recovery also handles the destruction of a critical file, such as the data base or the transaction queue.

Abort of a Single Transaction

If a user module employing I-D-S/II aborts while modifying the data base, the contents of the data base may be inaccurate. Should this condition occur, I-D-S/II, together with the TPCP, rolls back the data base (i.e., backs it up to reflect its contents before the transaction was processed) and reports the current transaction as failed to the TPCP.

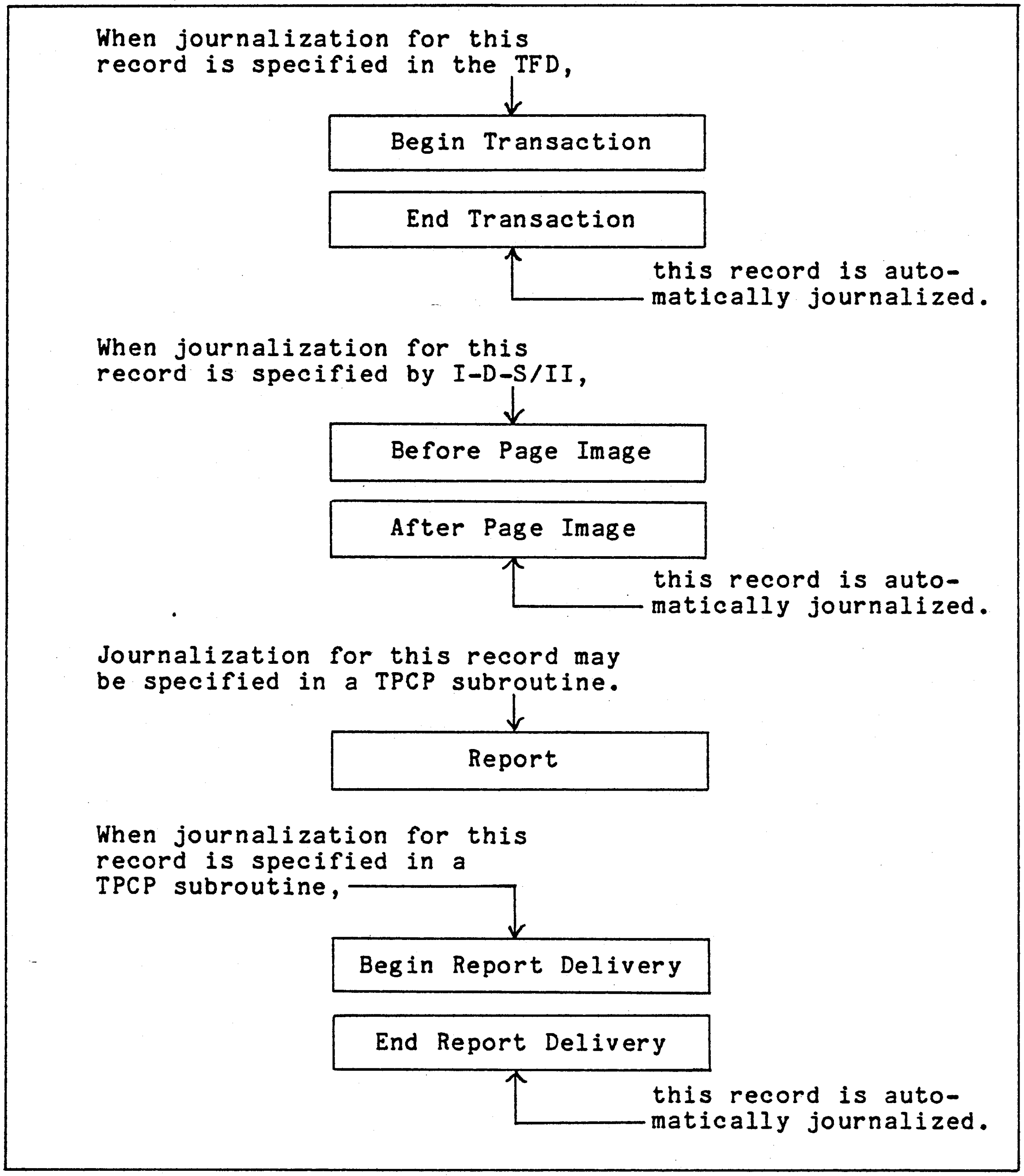
When journalization for this record is specified in the TFD,
Begin Transaction
End Transaction
this record is automatically journalized.
When journalization for this record is specified by I-D-S/II,
Before Page Image
After Page Image
this record is automatically journalized.
Journalization for this record may be specified in a TPCP subroutine.
Report
When journalization for this record is specified in a TPCP subroutine,
Begin Report Delivery
End Report Delivery
this record is automatically journalized.

Figure 14-2. Journalization Scheme

Table 14-1. Common Journal Records

Record	Definition
Begin Synchronization	Indicates that the data base area has been opened. I-D-S/II journalizes this record when opening an area (CP-6 random file) of the data base.
Begin Transaction	Indicates that the transaction has entered the system. The TP Command Processor journalizes this record before the transaction is queued. The TFD for the transaction type specifies whether or not transactions are to be journalized.
Before Page Image	Journalizes the data base page before the data base is modified. Journalization of this record is specified in I-D-S/II.
After Page Image	Journalizes the data base page after the data base has been modified. When Before Page Image records are journalized, After Page Image records are automatically journalized.
Report	Journalizes the report contents.
End Transaction	Proves that the transaction has been processed and indicates whether it was successful or had failed. When Begin Transaction records are journalized, End Transaction records are automatically journalized.
Begin Report Delivery	Indicates that the TP Command Processor is beginning to deliver the report or reports. Journalization of this record is specified in an application program.
End Report Delivery	Indicates that the TP Command Processor has completed delivery of the report. When Begin Report Delivery records are journalized, End Report Delivery records are automatically journalized.
End Synchronization	Indicates that the data base area has been closed.

Table 14-1. Common Journal Records (cont.)

Record	Definition
Crash	CP-6 recovery automatically writes this record after a system failure when closing journals.
TP Administrator End	The TP Administrator automatically writes this record to journals before closing them. This occurs whenever the system crashes.
Queuedump	This record is a snapshot of the transaction queue as it currently appears on disk. It is automatically written to the common journal when the TP Administrator opens the next tape in the series. When this record is journalized, together with its subsequent transaction records, reconstruction of the queue is expedited by limiting the number of journal entries to be scanned.
User	User records may appear on the journal. They will contain special header identification appended by the system.

System Crash

Events associated with the CP-6 monitor could also cause the system to crash. In this case, CP-6 recovery automatically checks all open files for incorrect linkages and inconsistencies. It records the cause of the crash, as well as other pertinent data, for subsequent analysis. Standard recovery can then be run.

Destruction of Critical Files

The data base or the transaction queue could be destroyed because of a hardware or a system malfunction or because of errors in user modules. The recovery process called by the TP Command Processor can reconstruct the data base and the transaction queue after failures within TP and CP-6.

ADDITIONAL FEATURES AND VARIATIONS IN SYSTEM USAGE

Protection and Control Features

TP provides extensive protection. User modules are written in the traditional manner and protection is provided centrally within the system. Some of the protection and control features are:

- Each user station is authorized by a log-on procedure in order to control access to TP components.
- A transaction may be journalized. This ensures that it will not be "lost" between its entry into the system and its processing.
- I-D-S/II may generate a record of the premodified data base so that changes made by an aborted transaction may be undone or a damaged data base may be restored.
- TP does not deliver reports until the transaction is successfully processed. Because reports and Begin and End Report Delivery records can be journalized, the loss of created - but undelivered - reports can be corrected.

System Queue Management Variations

CP-6 System queue management provides queue access services. Thus other types of jobs may avail themselves of these services and other processors may be substituted for, or added to, the TP system. For example, the queue could be employed by user programs executing under CP-6 time-sharing. It may be used for communication between two batch programs. It may also be used as a data collection medium where transactions are received at random intervals during the day, stored in the transaction queue, and processed during slow hours by a batch program similar to the a TP application module.

User-written Transaction Processing Modules

The programs which carry out the activities associated with processing each transaction are user-written and may be written in most CP-6 languages. however, COBOL is used most often. These modules may be developed and ckecked out the time-sharing mode using editing, debugging, and other facilities of the system.

Spawned Transactions

One transaction may generate other transactions. Transactions are spawned in user modules by calling a TPCP subroutine, which is typically used to spawn one or more transactions on a low priority basis. Spawned transactions break down a unit of work in order to improve response times. For example, in an inventory application, each time a part is removed from stock, the relevant user module decrements the number-in-stock item for that part. When the number-in-stock is reduced to the re-order point, a transaction is spawned to place an order for that part.

OPERATIONAL CONSIDERATIONS

Some operational features to be considered are

1. TP does not affect the availability of the CP-6 batch and time-sharing services. These services can be used concurrently with transaction processing. TP files are compatible with other CP-6 files.

2. TP provides a controlled interface between user modules and the I-D-S/II data base files accessed by those modules. It does not, however, provide this interface for any data base other than an I-D-S/II data base. The user module communicates with those files using the CP-6 file management system.

3. CP-6 handles file contention problems.

4. CP-6 enables the use of I-D-S/II as a public library capable of concurrently serving the data base access requirements of multiple on-line and batch users.

TRANSACTION PROCESSING TERMINALS

Terminal support is provided to TP through the general facilities of CP-6. All terminals supported for time-sharing are also supported for TP.

SECTION XVI

VIRTUAL MEMORY AND PROGRAM SECURITY

This section describes virtual memory and security and the impact of these features on shared processor facilities.

VIRTUAL MEMORY AND SECURITY

The Level 66 Virtual Memory and Security Option (VM&S) provides virtual memory capability and protection capability at the hardware level. The virtual memory feature is based on the concept of working spaces and segments within working spaces. The security aspect is grounded in this virtual memory management concept. The paragraphs that follow describe the CP-6 implementation of virtual memory and security.

The virtual memory concept provides for a large memory area which may be divided into 512 equal working spaces (WS) of 512K words each. A WS is further divided into variable size elements called segments. A segment has at least one descriptor that serves to locate the segment in virtual memory. The descriptor defines 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. The WS may be specified explicitly in the segment descriptor or indirectly through a Working Space Register whose number is contained in the segment descriptor. The hardware provides 8 Working Space Registers (WSR0...WSR7) that can be used to supply the working space number (0-511).

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 or execute).

An effective address is the address calculated by summing the address supplied in an instruction word, the address in a specified address register, and the address specified in an index register. The effective address is 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 equal parts of 1024 words called pages. Associated with each WS is a Page Table which identifies the physical pages allocated within the WS and the access that is to be 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 16-1.

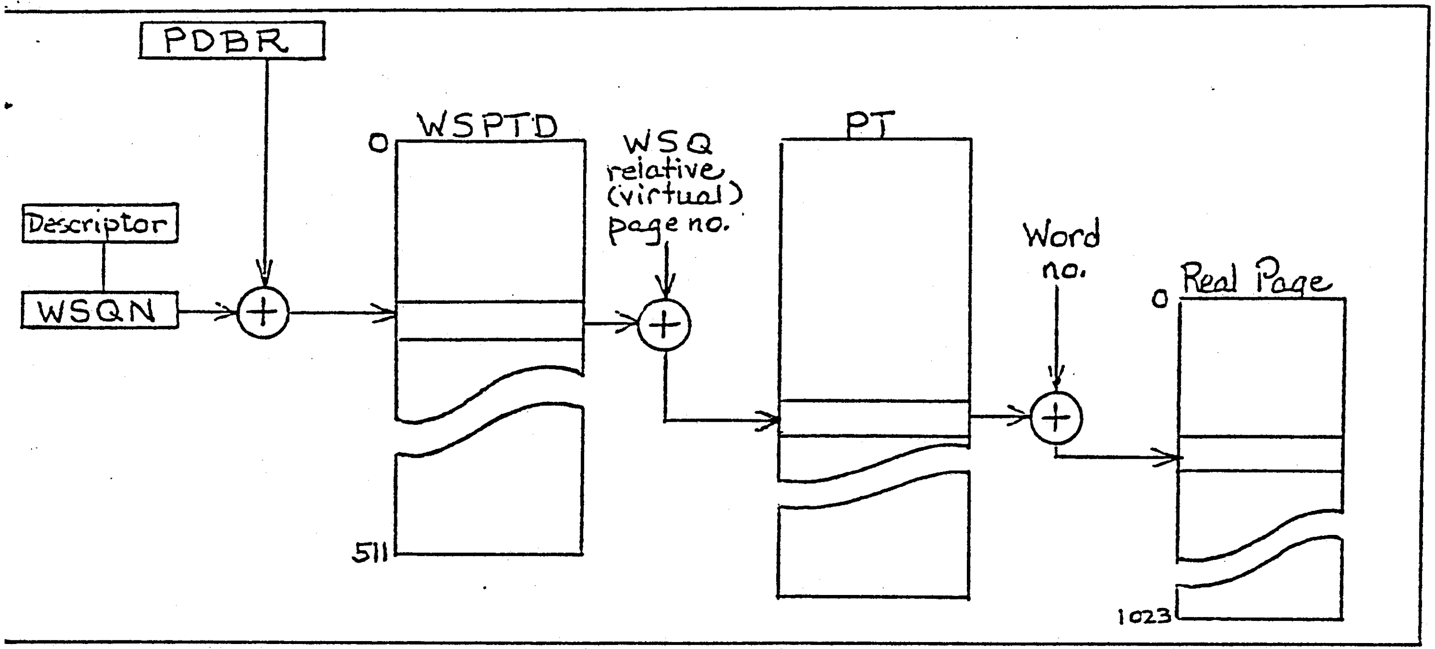

Figure 16-1. Memory Mapping

Another logical element of virtual memory is the domain. A domain is defined by the segments it may access and the access rights of those segments. The segments need not be contiguous and may encompass more than one working space. A domain may reference only those areas of virtual memory framed by the segment descriptors which are available to the domain. The operating system defines the user's domain by creating segment descriptors and passing them to the user. The user can not create a segment descriptor, or change the location or increase the size of the area originally framed by the monitor-prepared segment descriptors. Figure 16-2 shows two simple domains on a user's working space: that used by the user program and that used by the operating system.

A domain, then, is made up of segments. There are two general types of segments: Operand Segments which contain instructions, data, or a combination of both, and Descriptor Segments which may contain only descriptors. There are also two types of segment descriptors corresponding to the segment types.

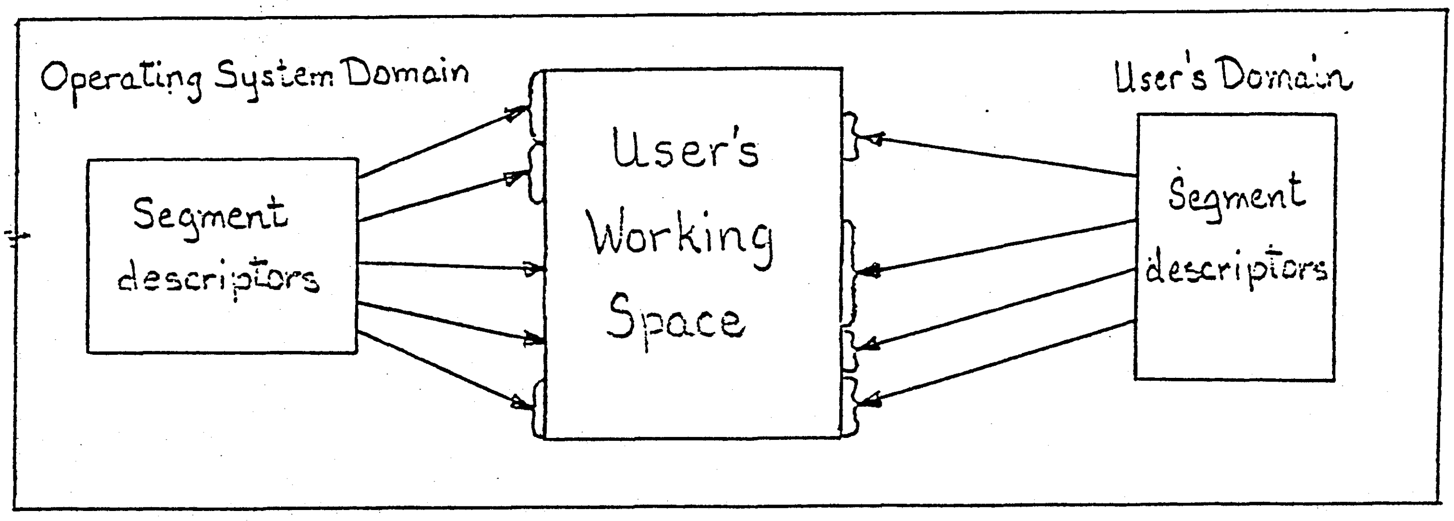

Figure 16-2. CP-6 Domains of Reference

There are two descriptor segments that are of particular interest. The first of these is the Linkage Segment (LS). The linkage segment is associated with a domain, and there is only one linkage segment for each domain. This segment is prepared by the operating system, is protected by the hardware, and provides the user program with descriptors that may be used to access its portion of virtual memory. The linkage segment is addressable through a special descriptor register, the Linkage Segment Register (LSR). The linkage segment and the LSR cannot be altered by a user program.

The other descriptor segment that is of particular interest is known as the Parameter Stack (PS). As with the linkage segment, there is only one parameter stack per domain. This descriptor storage is addressable through the Parameter Segment Register (PSR). The stack is empty until such time descriptors are entered during execution of the user program. This stack is used to contain descriptors that define parameters (areas of memory) to which another domain is to have access.

Operand segments are addressable through any one of nine Descriptor Registers (DR). The hardware provides a set of eight general operand descriptor registers (DR0...DR7) which may be loaded by the user to address any segment to which he has access.

The ninth DR is the Instruction Segment Register (ISR) which defines the domain's Instruction Segment. The ISR is associated with the Instruction Counter (IC) which locates the current instruction within the segment framed by the descriptor in the ISR. The ISR cannot be loaded directly. It is loaded by the hardware when control is transferred to a segment other than that framed by the current setting of the ISR.

In summary, the NSA hardware provides for several levels of isolation and protection. At the first level, everything is accessed via a descriptor which directly or indirectly (via a WSR) 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.

Another major factor contributing to the protection mechanism is that a slave mode user is prohibited from modifying any WSR, the LSR, and the contents of the Linkage Segment. The slave user is also prohibited from addressing in the absolute mode or manipulation of the page tables.

CP-6 supports five levels of execution for each user, each with its own set of privileges and visibility (areas of memory to which a process has access) - and with an established priority. Since each level of execution has its own Linkage Segment we can relate it to the NSA term "domain".

The domains supported by CP-6, i.e., the Linkage Segments the software will build in order of lowest to highest priority (and infallibility), are:

1) USR - User
2) ASL - Alternate Shared Library
3) IDB - Interactive Debugger
4) ICP - Interactive Command Processor
5) MON - Monitor

These five domains are shown in Figure 16-3, together with the paths of control transfer which exist between them.

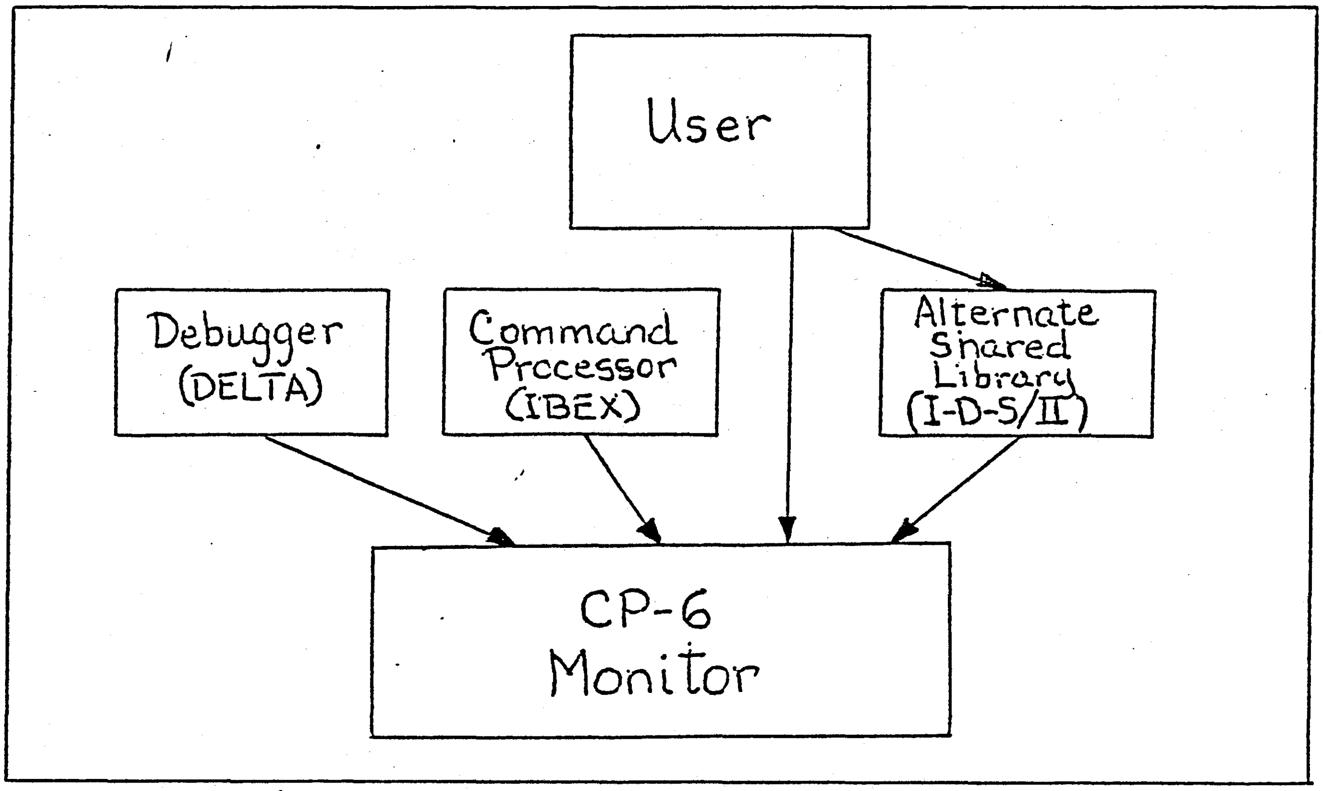

Figure 16-3. Control Paths Between CP-6 Working Spaces

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 VM&S option prevents any one user from accessing the memory of other users or special shared processors. The descriptors available to a given domain provide access to segments within the various domains only on a "need to know" basis. Even the monitor views only those parts of the user program which were designated on its call.

Change from one domain to another is effected by execution of the call form of the CLIMB instruction which establishes a new Linkage Segment and, usually, a new Parameter Segment. To call a new domain an Entry Descriptor is required. The Entry Descriptor describes the Linkage Segment that defines the called domain, the Instruction Segment of the called domain, and an entry point within the Instruction Segment.

The CLIMB instruction used to transfer control to another domain allows for saving 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 will restore the environment from SSS). The call form of CLIMB also provides for passing parameters to the called domain on the parameter stack which are removed by execution of the return form of CLIMB.

The domains of reference in CP-6 are summarized graphically in Figure 16-4. The linkage segments and parameter segments defining the domains of the debugger, the command processor (CP), the alternate shared library (ASL), and the user are physically contained in each user's working space HJIT. A descriptor of the user's JIT in each domain is not shown but is also present.

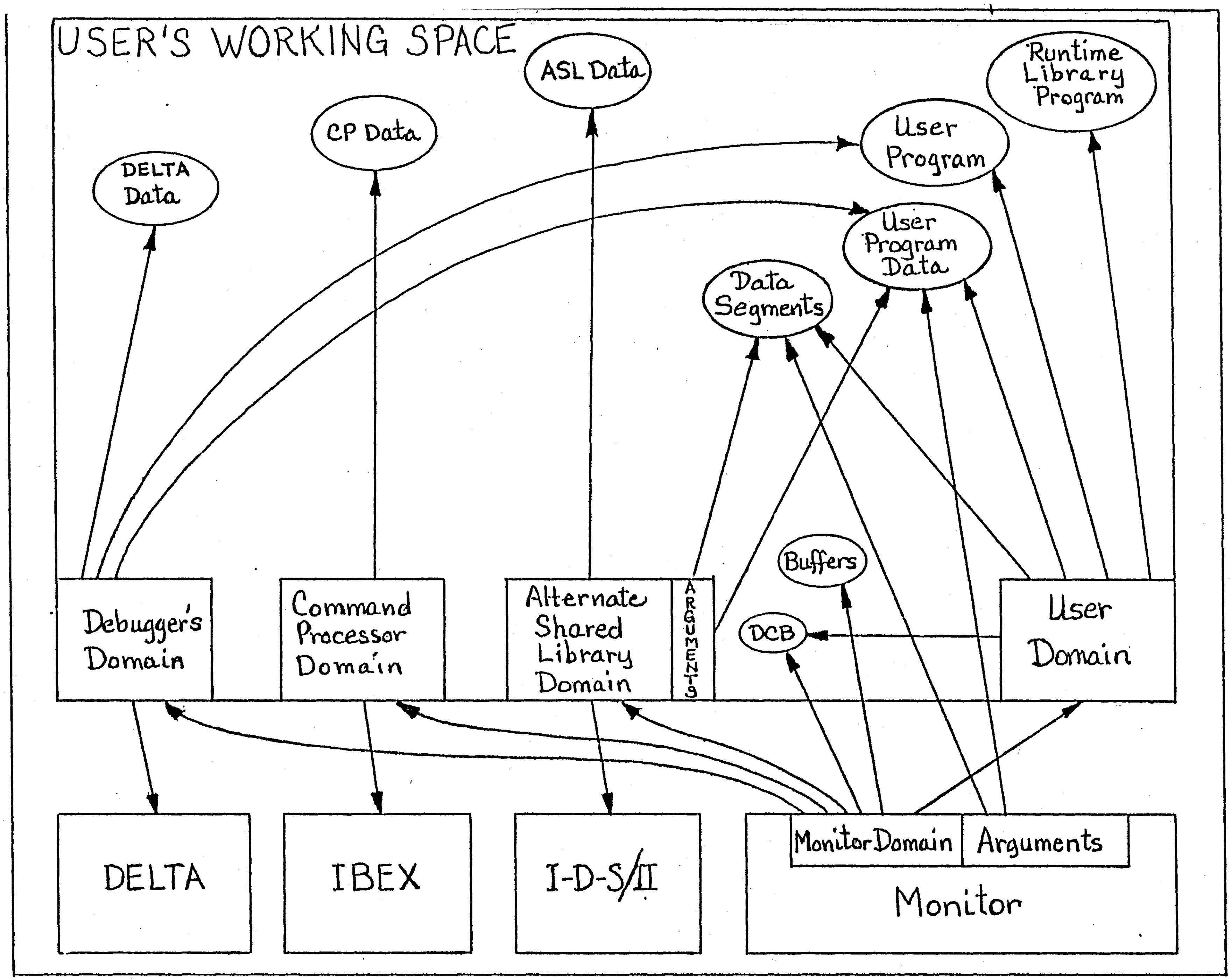

Figure 16-2. CP-6 Domains of Reference

The basic approach to memory allocation and execution control in CP-6 is as follows:

1) CP-6 uses only one 64-word Safe-Store Stack frame per domain.

2) 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.

3) Exit from a higher level domain restores the privilege and priority to those that existed prior to entry.

4) The monitor resides in its own working space.

5) Each user has a unique working space.

6) Up to three _special_ shared processors may be associated with any given user: an alternate shared library, a debugger, and the command processor. Each type of special shared processor resides in a separate working space.

7) Ordinary processors with shared procedure execute within the user's working space. These include most language processors such as APL, FORTRAN, BASIC, RPG-II, etc.

8) Working spaces are assigned to the following uses:

   - Real addressing. Can only be referenced by monitor procedure running in privileged master mode.

   - The monitor and its dynamic data segments, as seen by the monitor (all procedure on housekeeping pages).

   - Interactive command processors - procedure only.

   - Debuggers - procedure only.

   - Alternate shared libraries - procedure only.

   - The users. Each user's working space contains all pages that are assigned to him. Some of the pages in his working space may be shared: standard shared processor and run-time library.

User Virtual Memory Layout

The user's 512K words of virtual space is divided into the following segments:

1) Housekeeping JIT (HJIT)

   There are three segments of interest required by the NSA hardware which are provided for in the HJIT. These are:

   1) LS - the linkage segments for a user and all processes which may potentially be associated with a user.

   2) SSS - the safe-store stack.

   3) PS - the parameter stack.

2) Job Information Table (JIT)

   The JIT is divided into accounting information and the page table for the user's working space.

3) Buffers

   All file (and formerly COOP) buffers are allocated from a common pool within this segment.

4) Special Shared Processor Data Segments

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

5) Data Control Block (DCB)

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

6) The user's Instruction Segment (IS)

   This area provides a 256K 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.

7) User Data Segments

   This area can contain up to eight independent user data segments, the first two of which are used for PL-6 automatic data and for COMMON data.

The linkage segment provided to the user provides no access to the HJIT (other than via the LSR and PSR), the buffers, or Special Shared Processor Data Segments. The descriptors for the JIT and DCB segments allow read access only. The user has read, write and execute access via the descriptors for his Instruction Segment. The procedure portion within the Instruction Segment is protected at the page level. The descriptors for the User Data Segments allow read and write access.

Figure 16-5 shows the layout of the user's virtual space within his working space. With the exception of a fixed minimum requirement for HJIT, JIT, DCBs and buffers, the physical pages allowed to the user are demand allocated.

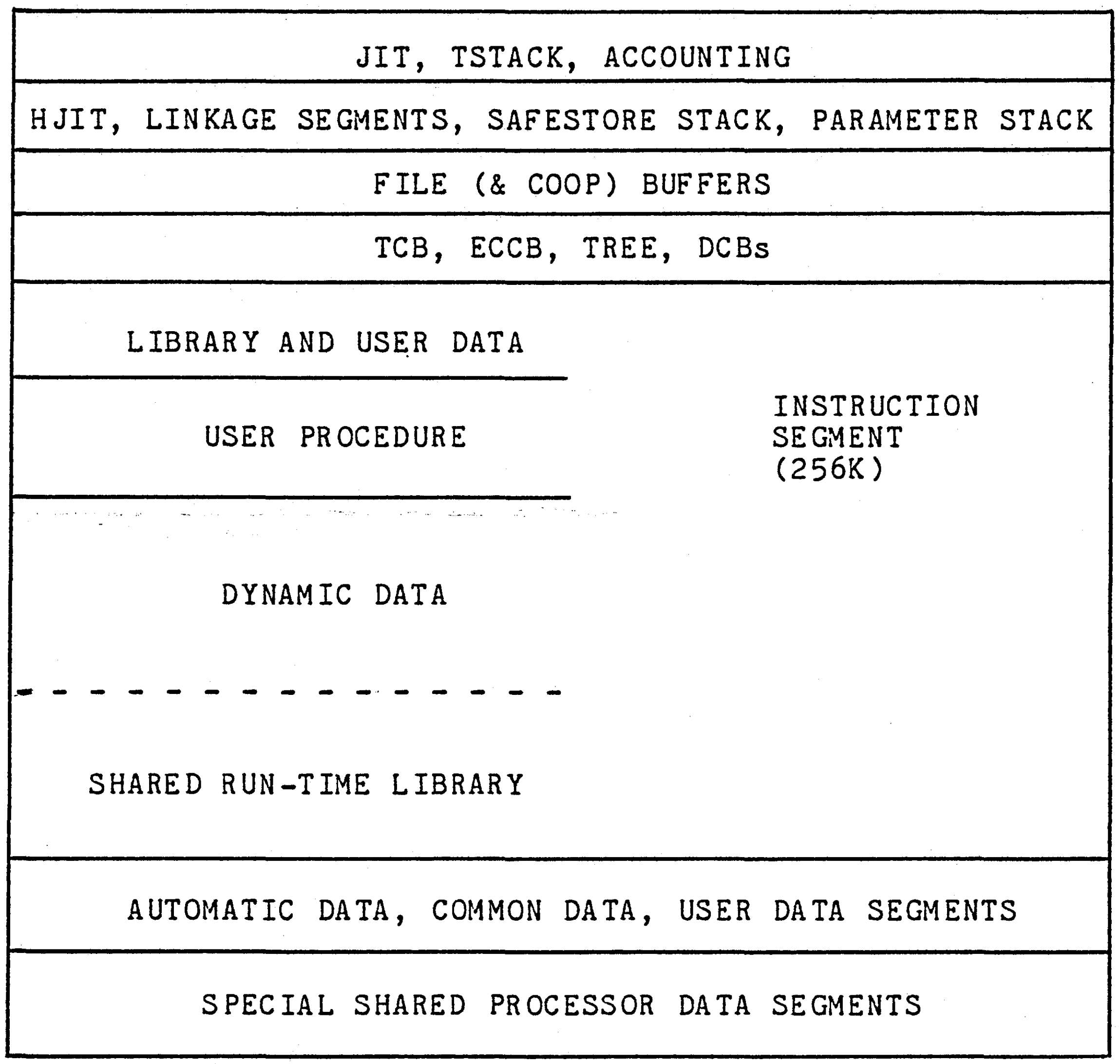
JIT, TSTACK, ACCOUNTING
HJIT, LINKAGE SEGMENTS, SAFESTORE STACK, PARAMETER STACK
FILE (& COOP) BUFFERS
TCB, ECCB, TREE, DCBs
LIBRARY AND USER DATA
USER PROCEDURE
INSTRUCTION SEGMENT (256K)
DYNAMIC DATA
SHARED RUN-TIME LIBRARY
AUTOMATIC DATA, COMMON DATA, USER DATA SEGMENTS
SPECIAL SHARED PROCESSOR DATA SEGMENTS

Figure 16-5. User Virtual Memory
512K Working Spaces (not to scale)

SHARED PROCESSOR FACILITIES

The shared processor facilities of CP-6 permit the sharing of the code for compilers, assemblers, command language processors, data base managers, debuggers, libraries, and other programs 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 processors are not limited to those offered under CP-6. The facilities may be effectively used whenever any program has a high probability of common usage. Service bureaus, for example, may use the mechanism for proprietary packages. Corporate installations may use the mechanism for programs with a high use frequency.

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

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

Figure 16-6 shows the relationship of the working spaces for CP-6 users and the working spaces for the special shared processors to real memory, thus showing the several ways that CP-6 shares program procedure.

Standard Shared Processors

Standard shared processors are created by the linker in the same manner as any user program. The processor resides in the user's working space. It may have initial data and DCBs which are unique for every user. The procedure portion of the processor is shared by all users associated with that processor by having the procedure portion mapped into each user's working space.

To qualify as a standard shared processor, a program must meet certain requirements. Two of the more important requirements are:

1) Shared processors are allowed only one level of overlay. There is no restriction on the number of overlays but only one of them can be associated at a time. In contrast with CP-V, data is allowed in overlays.

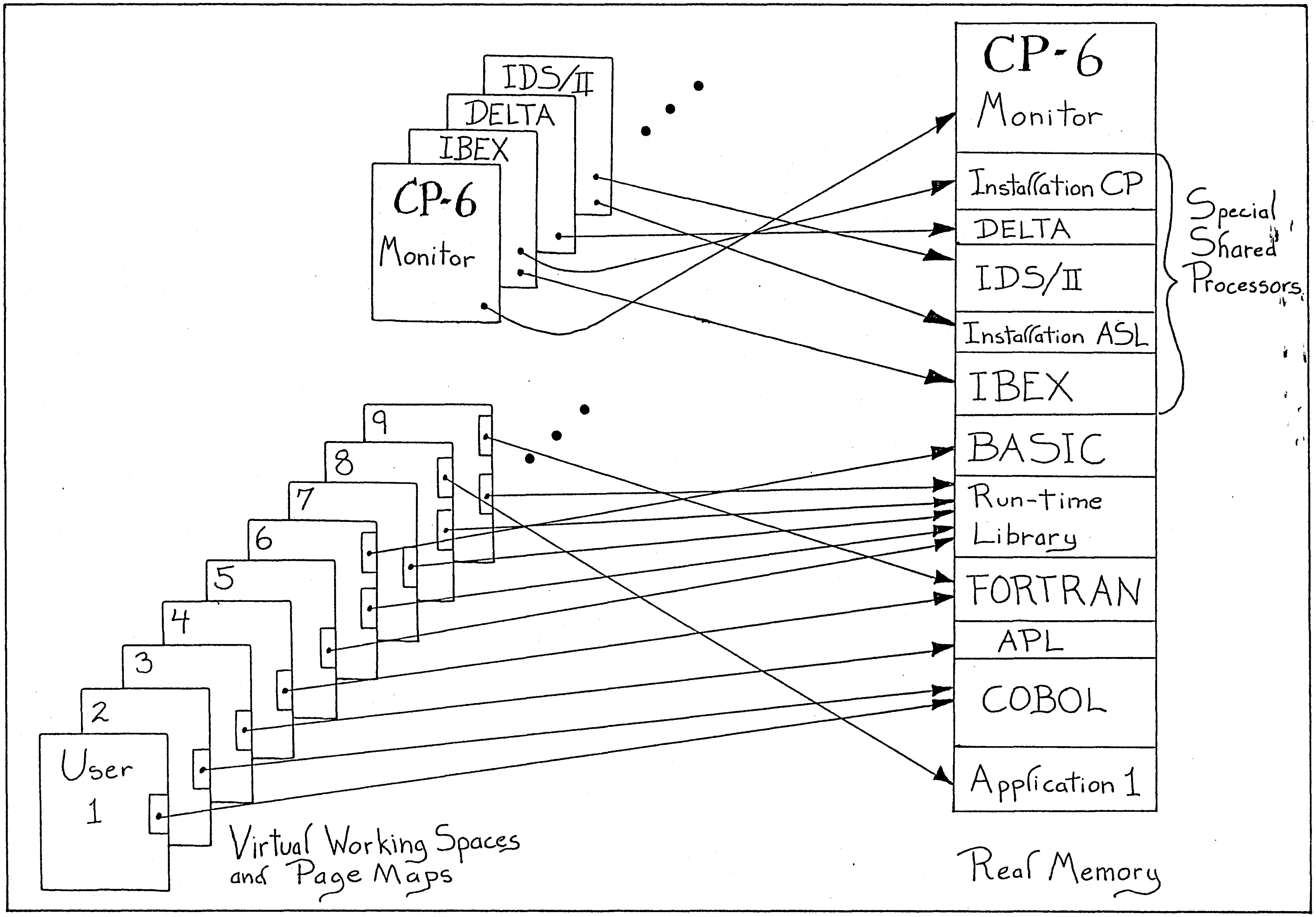

Figure 16-6 CP-6 Shared Processors (Processes)

2) Shared processors are often written in assembly language but may be written in FORTRAN or any processor providing pure code output. These shared processors must provide an initialization routine that associates any required run-time library and calls library initialization routines.

Shared Run-Time Libraries

A shared run-time library is a collection of frequently used subroutines which is treated by CP-6 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 CP-6 (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 Segments. 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

CP-6 recognizes three types of special shared processors:

- Alternate Shared Library
- Debugger
- Interactive Command Processor

The user program may be associated with one of each type of special shared processor at any given time.

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 command processor. It is both useful and common for installations to supply additional special shared processors in any of the three categories. Command processors are especially useful.

The procedure portion of a special shared processor resides in its own working space and 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 type of special shared processors was created to provide the I-D-S/II with an environment which 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 DCBs and buffers reside in the user's working space because they are unique to the user.

The alternate shared library has its own dynamic data segment fixed within the user working space to which the user has no access and this area is used to store the data base procedures, page buffers, subschema dictionary, and data base context. Alternate shared libraries will be given Exit Control (optionally) if the user aborts, giving I-D-S/II a chance to clean up the data base.

The descriptors in the Alternate Shared Library's Linkage Segment allow full access to the procedures in the alternate shared library's own working space and to the alternate shared library's data segments in the user's working space. The alternate shared library has read access to the JIT and DCBs within the user's working space, but has no access to the user's HJIT, Buffer Segment, Debugger Data Segments, Interactive Command Processor Data Segments, and the user's Instruction Segment and Data Segments (except via arguments that are passed when it is called).

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 Control Block posting are deferred while the library is in control.

SHARED DEBUGGERS

The CP-6 supplied debugger is known as DELTA. DELTA has a need to access everything the user has access to and must not be allowed to access procedure and data for other special shared processors. In addition, DELTA needs its own data area within the user's working space.

DELTA's procedure resides in 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, 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.

COMMAND PROCESSORS

The CP-6 supplied command processor is called IBEX (Interactive and Batch Executive). It is pure procedure that resides in the working space reserved for command processors. It requires only limited access to users' working spaces, JIT, DCBs, and command processor data segments. In addition to processing all commands (see Section 7), 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. Because only one command processor can be associated with a given user at a given point in time, and the virtual address range of a working space is large (up to 16 million words), no conceptual problems exist.

SECTION XVII

SYSTEM PROGRAMMER FACILITIES

INTRODUCTION

This section provides an overview of those CP-6 features that are designed to aid the systems programmer in the development and maintenance of shared entities. For readers not directly involved in the development or maintenance of CP-6, it is recommended that this section be omitted because of its level of detail.

A large portion of the operating system and its associated processors is coded in PL-6, a high-level language developed especially for the implementation of CP-6. Some portions of the system are implemented in assembly language. All specific instructions mentioned within this section are assembly instructions.

SHARED ENTITIES

Five kinds of shared entities are discussed:

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

Shared libraries reside in the user's working space at a fixed origin within his ISR segment (see Figure 17-1). Library data is always at a second fixed origin (zero) within the ISR segment. Standard linkage to shared libraries utilizes the direct subroutine branch (TSX) instruction.

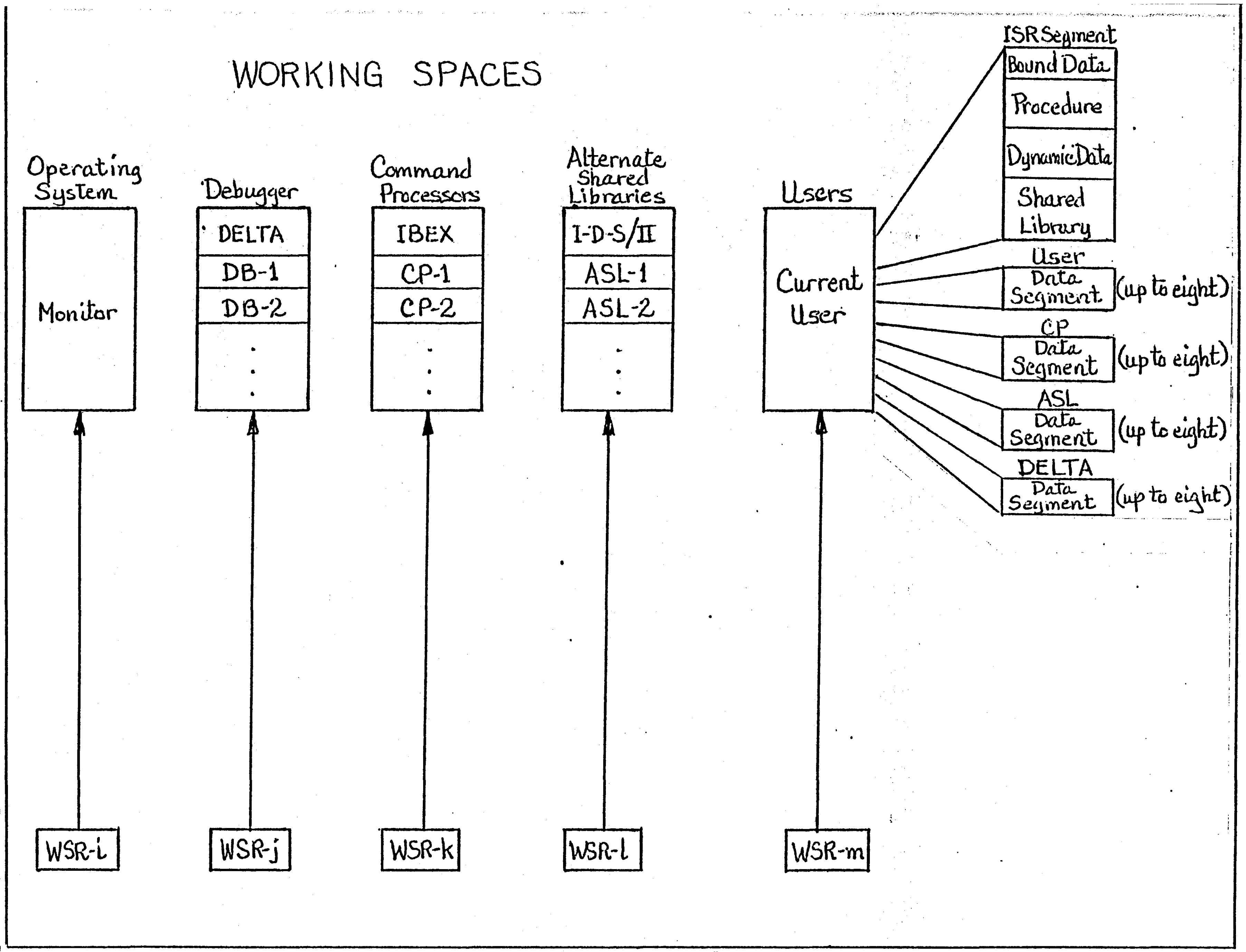

Figure 17-1. CP-6 Context for Sharing

Shared processors run as user programs in the user's working space utilizing dynamic data and user data segment space as required (see Figure 17-1).

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 which are reserved for command processor (CP) use (see Figure 17-1). 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 (in the user HJIT). There is no direct linkage (CALL-RETURN) between command processors and user programs.

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, ASLs obtain data space from the user's working space data segments which are reserved for ASL use (see Figure 17-1), and the ASL linkage segment is in the user's HJIT. Linkage from user to ASL is via the CALL form of CLIMB (inward CLIMB) with parameters passed on the Parameter Stack. Exit from ASL to user is via RET (outward CLIMB).

The DELTA debugger's procedure resides in its own working space and thereby 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 to which the user has access, 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, it is entered for the user by the monitor 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.)

The monitor resides in its own working space (see Figure 17-1). Requests for monitor services are via the PMME form of CLIMB with parameters passed on the Parameter Stack. Normal return is via an outward CLIMB.

MEMORY MANAGEMENT

The following provides a brief outline of memory management restrictions and services.

1. Overlays

   Shared libraries, command processors, and alternate shared libraries may not be overlaid; shared processors are restricted to one level of overlay. This overlay level may have data (which is an extension over CP-V facilities).

2. Dynamic data

   Monitor services exist to get and free dynamic data pages on a "next-last" basis (M$GDP-M$FDP) or on a virtual page number basis (M$GVP-M$FVP).

3. Data segments

   Monitor services exist to allocate and expand (M$GDS) and to contract and free (M$FDS) data segment space. Based on the domain of the caller (i.e., the current linkage segment) data segments unique to the user, command processor, or alternate shared library are manipulated.

4. Services for processors with special privileges

   A set of services exist to acquire real pages, map them into virtual pages and manipulate access control flags for virtual pages. These are available only to processors with appropriate privilege levels.

INTERFACES

In CP-6, all program 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 service calls. For these service calls, transfer of control is effected via the PMME form of CLIMB with XO specifying both the kind of service requested and whether or not errors in processing are to be signaled by means of an alternate return from the monitor. Part of the calling sequence specifies the location of an FPT (Function Parameter Table) which contains information pertinent to the service request (parameters) as well as information pertinent to how results - if any - are to be returned.

EXCEPTIONAL CONDITION HANDLING

Facilities are provided whereby programs can specify parameterless procedures to be 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 whence the trap occurred, transfer to a previously specified label (unwinding automatic storage in the process), or merely pop the frame on the condition stack.

ACCESSING SHARED ENTITIES

Centralized enqueue/dequeue facilities are provided for general use in controlling access to anything that is shareable by more than one program: data, files, other programs, etc.

INITIALIZATION AND DATA

Of the five shared entities discussed in this chapter - shared libraries, shared processors, command processors, shared debuggers, and alternate shared libraries - only the first two have any data space with which to work when they are first entered, and only shared processors may have initialized data (just as any user program may). Command processors, debuggers, and alternate shared libraries must "bootstrap" themselves data space by first making a monitor service call (with a constant FPT) requesting a small amount of data segment space. Such a service call cannot request that results be returned. This acquired space can then be used to make "full blown" requests for space.

STAR FILES

There is one type of file in CP-6 whose names are guaranteed by the system to be unique for a given job (or terminal user): the star file. The uniqueness is ensured by two mechanisms: (1) A file is a star file if, and only if, the first character of its name is an asterisk (*); (2) The directory for star files is named *T and is itself catalogued in 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 file names are reserved:

*T	-	the directory for star file
*A	-	assign/merge information
*S	-	step accounting
*G	-	object unit output from compilers
*L	-	default load module
*N	-	used by LDLNK

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 thru the M$SI DCB. The important standard DCBs are:

M$CI	Compressed input
M$SI	Source input
M$DO	Diagnostic output
M$LO	Listing output
M$SO	Source output (which is optionally compressed)
M$GO	Binary output
M$UC	User terminal (for interactive I/O)

In CP-6, use of the full capabilities of IBEX, especially those relating to default assignments, requires adherence to these conventions.

SYSTEM FACILITIES

CP-6 offers a number of centralized services to aid the systems programmer. Two of them are described here.

- A common command parser exists for use by shared entities. It accepts as input a pointer to a text string and a table describing commands and options to be searched for and produces as output a parsed tree of the input text.

- Another system service interprets the text string input to it as a file identification and outputs the data required by M$OPEN FPT.

- A centralized set of error message files are supported for use of the operating system as well as user programs. They are keyed by functional code group, module identification, error code, and severity level. A centralized service to access a given record in the files and print its content (with parameter substitution) is provided.

STANDARDS AND CONVENTIONS

CP-6 has a number of support programs and services which are based on the standards and conventions adopted for system implementation. To the extent that users adhere to the same standards and conventions, the support programs and services can be utilized. One example - the centralized error message files - has already been cited. The standard key is a two character functional code group, a one character module ID, a fifteen bit error code, and a 3 bit severity level. The error message print service utilizes a user-supplied file name (or defaults to the standard system file) together with this key to access the message. The same service is utilized for HELP facilities. Support programs for maintaining error message files are provided. Another set of conventions deals with use of PL-6 comments in standard fashion so that an EXTRACT program can produce "automatic" system technical documentation.

COMMON LIBRARY

A comprehensive library of subroutines commonly used by applications and systems programmers is available for use both for providing individual subroutines and for incorporation into shared libraries. Among other things, it includes:

- receiving sequence and automatic storage handling routines.

- binary, complex, decimal and double precision arithmetic.

- numeric and alphanumeric comparison.

- type conversion.

SECTION XVIII

SYSTEM MANAGEMENT FACILITIES

INTRODUCTION

The manager of a CP-6 system may exercise considerable control over the system through use of the following facilities:

- System Definition
- System Performance Control
- Resource Management
- User Authorization
- Use Accounting

SYSTEM DEFINITION

CP-6 system definition facilities include the programs and procedures in the system which accomplish the following:

- Configuration and Reconfiguration - tailoring the system to the particular hardware configuration. Parts of this process occur during cold start, parts during warm start or recovery, and parts are controlled dynamically and thus may be changed during system operation.

- System preparation - compiling and loading parts or all of the system. In CP-V, this process was referred to as SYSGEN.

Much more of CP-6 is defined at run-time or determined at system start-up than was the case with CP-V. It is easier to define a CP-6 system than it was to 'SYSGEN' a CP-V system. In particular, the start-up or "boot time" adaptation of the system to its hardware includes generation of tables for peripheral devices, I/O enqueueing, physical memory management, and managing the users of CP-6.

Certain tables grow and shrink dynamically, adapting to the demands of run-time. This relieves the system manager or analyst of the difficult chore of "pre-guessing" the requirements. Tables handled in this way are:

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

As in CP-V, CP-6 carries definitions of remote processing workstations in files that may be created and altered during system operation.

SYSTEM PERFORMANCE CONTROL

Batch Stream Control

The system manager is able to allocate the resources of the system to jobs with certain attributes by defining a set of batch streams under which diverse categories of jobs may run. A batch stream is a collection of ranges of job attibutes. In effect, it is a job profile. Physical system resources such as memory, spindles, or tape drives are not permanently allocated to a particular batch stream. All jobs executing in the various batch streams draw their physical resource requirements from a common pool without regard to the stream under which they qualified for execution except that the numeric limits that pertain to that stream will apply. Examples of attributes that comprise a batch stream 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 entering CP-6 for batch execution share the same input queue (referred to as the batch job queue). Jobs are selected from this queue for execution in the batch streams.

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 stream tables are searched for a stream that fits the requested resources and is currently available.

4. If a stream is not available for the selected job, the next job is considered as in steps #1, 2, and 3.

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

It is recommended that at least 32 batch stream definitions be generated for all systems unless main memory is a serious consideration. This will provide a variety of job attribute classifications and those batch streams in excess of the operational number may be locked from use through a processor (CONTROL) available to the system manager.

In a time-sharing/batch processing system, emphasis may be given to batch processing by opening up more batch streams. However, it should be noted that CP-6 is a queue-driven system and tasks are selected from prioritized queues without regard to the source of the request (i.e., on-line, batch, or remote batch). When there is a heavy on-line user load, as the number of batch streams increases, the number of compute-bound tasks increases and each receives a small fraction of the CPU time. This means that batch jobs will be able to get more CPU time because of large quanta assigned to the batch streams. This will not make a significant difference in on-line response time because interactive requests have a higher priority than compute bound jobs. More attention may be given to certain categories of batch jobs by increasing the number of batch streams suitable for them.

System Tuning and Measurement

CP-6 has 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 three processors that provide these facilities are

1. CONTROL
2. STATS
3. SUMMARY

The CONTROL processor provides a means of adjusting system performance. Commands of the CONTROL processor enable the system manager to display certain 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 stream definitions.

The STATS processor performs two functions. One function consists of displaying selected performance data in real-time. The other function consists of creating "snapshot" records of performance data for later processing by the SUMMARY processor.

The SUMMARY processor provides a global view of system performance by formatting and displaying the statistical data collected by STATS. The input data for SUMMARY is the SNAPSHOT file created by STATS. The output listings are generally large and therefore must be output to a file or on the line printer.

The SUMMARY processor allows the system manager to

1. Request a chronological listing of snapshot data for one or more display groups.
2. Specify a sort filter to remove undesired snapshots from the sample for subsequent reports.
3. Request filtered, sorted, and ordered listings of snapshot data for one or more display groups.
4. Request filtered, sorted, ordered, and averaged listings of snapshot data for one or more display groups.

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 a manner such that each user requiring the resource is allocated the resource for its exclusive use. (An exception to this is private disk packs which, under some circumstances, may be shared even though they have been defined to be resources.) Peripheral devices, CPU time, 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.

There are special resource management routines within the monitor. The specific task of these routines is to keep track of the number of resources of each kind in use and 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. Further, the resources are reserved for the exclusive use of the job. Thus it is guaranteed that they will be available even if time elapses between job start-up and actual use of the resources.

CP-6 does not require that an actual physical device correspond to each of the resources it manages. 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.

The system manager must define what the resources are for the installation, establish system defaults and maximums for use of the resources, and set limits on the use of the resources for the individual users. He performs these tasks using the following processors:

- System definition
- CONTROL
- SUPER

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

The CONTROL processor can be used to dynamically modify the default and maximum values associated with each resource. Resources are defined during system definition, but 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.

In order to coordinate the sharing of a CP-6 installation among many users, it is necessary to impose resource limitations on the execution of user programs. When a job is started, limit values for the job are initially set from the :USERS file record. Values which are not specified in that record are then set from the monitor limit tables. For batch jobs, limit values are set to the values specified by an (optional) LIMIT control command. (A new LIMIT command may be issued at any job step so long as the limits specified are less than those previously specified.)

Finally, these composite values are compared to the maximum values in the :USERS file record or monitor limit tables and the job is aborted if the limits are exceeded.

USER AUTHORIZATION

User authorization is achieved in the following way. Before any user can do any processing on CP-6, an account must be created for him 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 following additional parameters may be supplied 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:

   - Utilize real-time services.
   - Access and change the monitor.
   - Read and write error file; request the devices; invoke diagnostics; authorize enqueue/dequeue automatically.
   - Examine (but not change) the monitor.

2. The password that is to be associated with the account. If specified, no one can do any processing unless the password is provided.

3. Whether or not all files created under this account may be read, executed, or modified by other users. There is a default which applies to files created by the user unless the user explicitly gives overriding instructions.

4. Whether or not 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 or not the processors available to this account should be restricted.

6. Whether or not 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, with the system manager's discretion, be applied to users on an individual account basis.

USE ACCOUNTING

During the operation of each job, CP-6 accumulates a wide assortment of counts which record the job's activities. These include CPU use, memory use, I/O operations, pages printed, cards punched, monitor service requests, terminal I/O character rates, and many others. These counts are written into a file which may be used by the installation to prepare charges for its customers. Hooks are provided so that installation-supplied routines may augment or modify the records written to the accounting file. Further, 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 program executing. This makes charging for proprietary program products especially easy.

For installations that wish to use it, a program called RATES is provided. RATES allows the installation manager to define monetary charges for each of the system counted values. Given these values, the system will automatically calculate the proper charge for the user session or job step. Additional features allow for 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 'hooks' within the charge calculating program provide for 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 RATE processor was used and details at each job step are optionally available.

Budget accounting is also available in CP-6. This permits an installation to establish a budget hierarchy and to control access to the system depending on a user's remaining budget. If a user or anyone above him has exhausted the budget, he may or may not be denied access to the system. 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 or may not be the last one the user is allowed.

SECTION XIX

COMPUTER OPERATIONS

INTRODUCTION

The computer operations facilities for CP-6 are still in the design phase. This section would be empty except that it was decided to include a description of the CP-V computer operations facilities since CP-6 is an outgrowth of CP-V.

SYSTEM START-UP AND 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 whenever possible--generally requiring no operator intervention.

JOB AND SYSTEM CONTROLS

The operator controls system operation through the use of console key-ins. 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). CP-V key-ins are listed in Table 19-1.

Table 19-1. Operator Key-ins

Key-In	Function
ABORT	Abort user or job.
ANSMOUNT	Inform monitor that an ANS tape has been mounted.
ANSSCRATCH	Inform monitor that an ANS scratch tape has been mounted.

Table 19-1. Operator Key-ins (cont.)

Key-In	Function
D	Enter date.
DATE	Enter date.
DELETE	Delete symbiont file from system.
DIAG	Authorize customer engineers to run diagnostics.
DISPLAY	Send system information to operator.
E	Error (terminate) job step - go to next job step.
ERROR	Error (terminate) job step - go to next job step.
ERSEND	Build a record in the system error log file.
FLUSH	Delete concurrent mode output being generated by a specified job for a specified device.
FORM	Change the form name on output files in the system.
GJOB	Initiate a ghost job.
HEADING	Provide message for on-line top-of-page heading or cancel previous heading.
INT	Transfer control to user's console interrupt routine.
MOUNT	Inform monitor that tape or pack is mounted.

Table 19-1. Operator Key-ins (cont.)

Key-In	Function
OBOFF	Disallow entry of jobs to the batch stream from on-line terminals and processors.
OBON	Reallow on-line and processor entry of jobs to the batch stream after an OBOFF key-in.
OFF	Allow no more users to log on.
ON	Set maximum number of on-line users.
ONB	Set maximum number of batch users.
OUTPUT	Place all output streams of a job into concurrent output mode or release a device from the concurrent output mode.
OVER	Override the rejection of an output tape. (Applicable only for ANS tapes in the semi-protective mode.)
PRIORITY	Change user file or execution priority.
RBBDCST	Add message to the remote message file.
RBDISC	Disconnect a remote processing terminal.
RBLOG	Allow automatic log-on of a remote processing terminal.
RBS	Allow connection of remote processing terminal.
RBSEND	Send a message to a remote terminal.

Table 19-1. Operator Key-ins (cont.)

Key-In	Function
RBSWITCH	Switch output files from one workstation to another.
RBX	Disconnect (and disallow connection of) one or all remote processing terminals.
READ	Inform the monitor that a tape without a write ring (for which the user specified that both reading and writing would be done) will be read only.
REQUEST	Prepare to dismount tape from a specified unit or request the tape type of a specified resource.
SCPU	Start the specified secondary CPU.
SCRATCH	Use the specified tape as a scratch unit.
SEND	Issue message to a specific on-line user or to all on-line users.
SS	Start symbiont card reader.
START	Search for input symbiont files to run.
S	Initiate symbiont action.
T	Enter time.
TIME	Enter time.
X	Abort user or job.

Table 19-1. Operator Key-ins (cont.)

Key-In	Function
ZAP	Abort all users and save the symbiont pointers for restart.
device,action	Initiate action indicated on the specified device (in response to a previous device message).

REMOVABLE STORAGE INITIALIZATION

INITVOL initializes pack sets for use with the file management system. It is used to establish serial numbers, account directories, and granule authorization and to write headers and other system information on selected areas of the volumes.

The LABEL processor initializes ANS tapes by writing ANS formatted labels. In the ANS protective mode, all ANS tapes must be prelabeled by LABEL. In the semi-protective mode, ANS tapes may be prelabeled by LABEL or may be given ANS labels as the result of an operator key-in.

PERIPHERAL DEVICE ERROR PROCEDURES

If the monitor encounters an abnormal condition during an I/O operation, it will send a message (Table 19-2) to the operator. These device error 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 key-in

```
device,action
```

where 'device' specifies the device and the 'action' is one of the following:

C	continue as is.
E	continue but inform the program of the error.
R	retry I/O operation, possibly after correcting the problem (e.g., by moving the error card back to the read station).

Table 19-2. Device Error Messages and Operator Action

Message	Operator Action
device MANUAL	Ready the device.
device WRITE PROTECT	Error (E) or remove write-protect and retry (R).
device TIMED OUT	Retry (R) or error (E). Time-out values are measured in ticks of a 5-second clock. 1. Tape rewind and space file - 50 ticks. 2. Operator terminal input - 100 ticks. 3. All others - 2 ticks.
device ERROR	Retry (R), continue (C), or error (E) if card reader or line printer; otherwise the error is irrecoverable and no operator action is needed or possible.
device NOT OPERATIONAL	Device busy, not recognized, or I/O not accepted. Correct the condition, usually dial tape unit or turn power on, and error (E) or retry (R).

If a required device is in manual status, the following message is typed every 20 seconds:

device MANUAL

In all other cases, if an operator action is required and none is received, the following message is issued:

device PLEASE RESPOND

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.

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 allows an installation to have selected periods of time (for example, graveyard shift) to provide time-sharing or 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 some on-line or batch job happens to request 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.

SECTION XX

RECOVERY

CP-6 attempts to make the system available as much as possible with minimal loss of data when problems occur. To this end, a recovery package is available which takes actions based on the seriousness of any problem that occurs. CP-6 accomplishes the recovery completely automatically, requiring minimal operator intervention only for the most serious problems, such as power interruption.

The various modules of CP-6 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 is judged to have compromised the integrity of the resident operating system, recovery is initiated and one of three 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. This involves writing the contents of main memory to secondary storage, writing out updated file buffers for the user, and eliminating the user job. The system is then allowed to proceed for all other users.

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 Normal Recovery is performed. The memory image is written to secondary storage. The context for each user is then examined. All open files are closed with default options. Remaining input for batch jobs which 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 brought in from the system device. Before resuming normal operation, accounting records are written. At this point, normal system operation proceeds. Terminal users must log on again.

3. If the key system tables are damaged, a procedure called Extended Recovery is performed. The memory image is written to secondary storage. Each individual file in the system is then examined for space

allocation information. The allocation tables are rebuilt and dual allocations (i.e., situations in which more than one file is trying to use the same space in the file system) are noted. When this process is complete, the system is reinitiated.

After any of the three types of recovery has been performed, the monitor dump analysis 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 and the status of the CPUs at the time of the problem.

SECTION XXI

HARDWARE MAINTENANCE AND THE DIAGNOSTIC SYSTEM

SYSTEM ERROR LOG FILE

All hardware malfunctions and some software problems occurring during system operation, whether recovered or not, are recorded in a special disk storage file. This file is periodically copied into a standard file by a program which is initiated automatically for that purpose.

The error file may be listed and summarized by the error log listing processor. The error file is also available for on-line preventive maintenance of the system and for diagnosis and prediction of hardware malfunctions.

Not all error file records are the result of error conditions. For example, a time stamp record is entered once each hour on the hour and an I/O activity count is recorded each hour and at recovery.

The error log listing processor provides an efficient tool for listing and sorting the error log file. Its output furnishes a meaningful, comprehensive diagnostic evaluation of the system and its peripherals, aiding in the early detection of product failures and thus increasing the reliability, maintainability, and availability of the system.

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 DIAGNOSTIC FACILITIES

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

APPENDIX A

CP-6 HARDWARE

The equipment supported by CP-6 is listed in Table A-1.

Table A-1. CP-6 Hardware

Equipment	Number Supported
CPUs (several models)	up to four
Main memory	many megawords
I/O multiplexers	up to 4
Disks with single or dual access	
100 mb removable	up to 32 per controller
200 mb removable	up to 32 per controller
Tapes with single or dual access	
9 track 800/1600 bpi 75/125/200 ips	up to 16 per controller
Unit record devices	
Card reader 1050 cpm	several
Card punch 100-400 cpm	several
Line printers 1200/1600 lpm (64 or 96 char)	several
Operator consoles	several
Local communications processors (CP)	up to 4 (or more)
Local real-time processors	up to 4 (or more)
Remote communication processors	up to 16 (or more)
Remote real-time processors	up to 16 (or more)

Table A-1. CP-6 Hardware (cont.)

Equipment	Number Supported
Communication lines per local or remote communication processor (asynchronous/synchronous)	up to 128
Level 6 peripherals	several
Printer	
60 lpm 64/96 character serial	several
300 lpm 64/96 character line	several
600 lpm 64/96 character line	several
Card reader 300/500 cpm	several
Cartridge disk 2.5-10 mb fixed and removable	several
Diskette 256 K bytes	several
Magnetic tape 9T 800 bpi 75 ips	several

APPENDIX B

CP-6 SOFTWARE MANUALS

The entire set of CP-6 software documentation is listed below.

- CP-6 Concepts and Facilities
- CP-6 Programmer's Reference Manual
- CP-6 Programmer's Guide
- CP-6 Monitor Services Reference Manual
- CP-6 Operations Reference Manual
- CP-6 System Management Reference Manual
- CP-6 System Programmer's Reference Manual
- FORTRAN Language Reference Manual
- CP-6 FORTRAN User's Guide
- CP-6 DELTA Reference Manual
- APL Language Reference Manual
- BASIC Language Reference Manual
- CP-6 TEXT Language Reference Manual
- CP-6 TEXT User's Guide
- RPG-II Language Reference Manual
- IDP Language Reference Manual
- CP-6 IDP User's Guide
- CP-6 Assembler Language Reference Manual
- SORT/MERGE Language Reference Manual
- COBOL Language Reference Manual

- CP-6 COBOL User's Guide
- I-D-S/II Data Base Administrator's Guide
- I-D-S/II Programmer's Guide
- CP-6 I-D-S/II User's Guide
- PL/I Language Reference Manual
- CP-6 PL/I User's Guide
- CP-6 Common Index
- CP-6 Pocket Guide
- CP-6 Conversion Manual
- CP-6 Real-Time Reference Manual
- CP-6 Remote Processing Reference Manual
- CP-6 PL-6 Language Reference Manual
- CP-6 PL-6 User's Guide