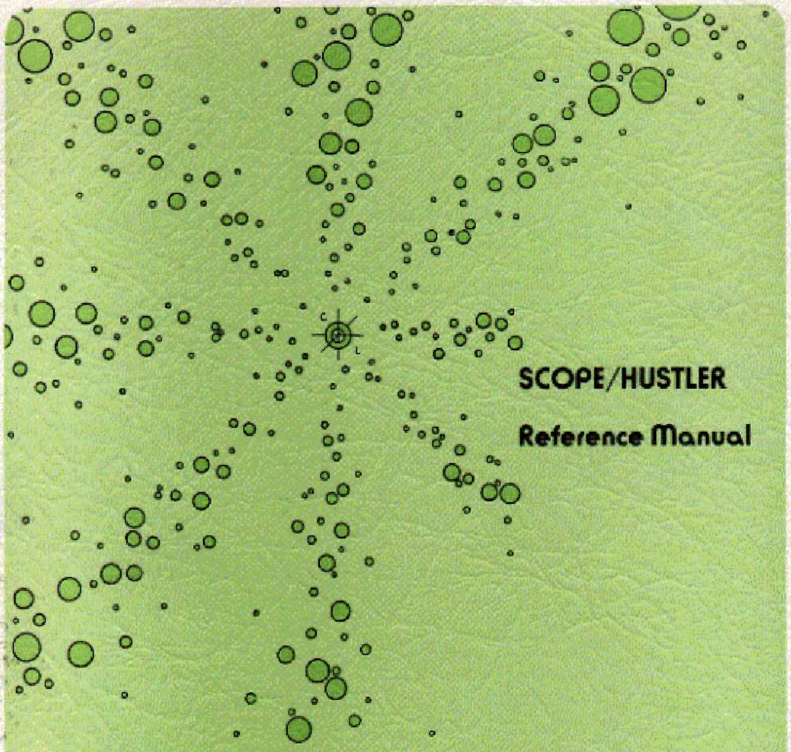
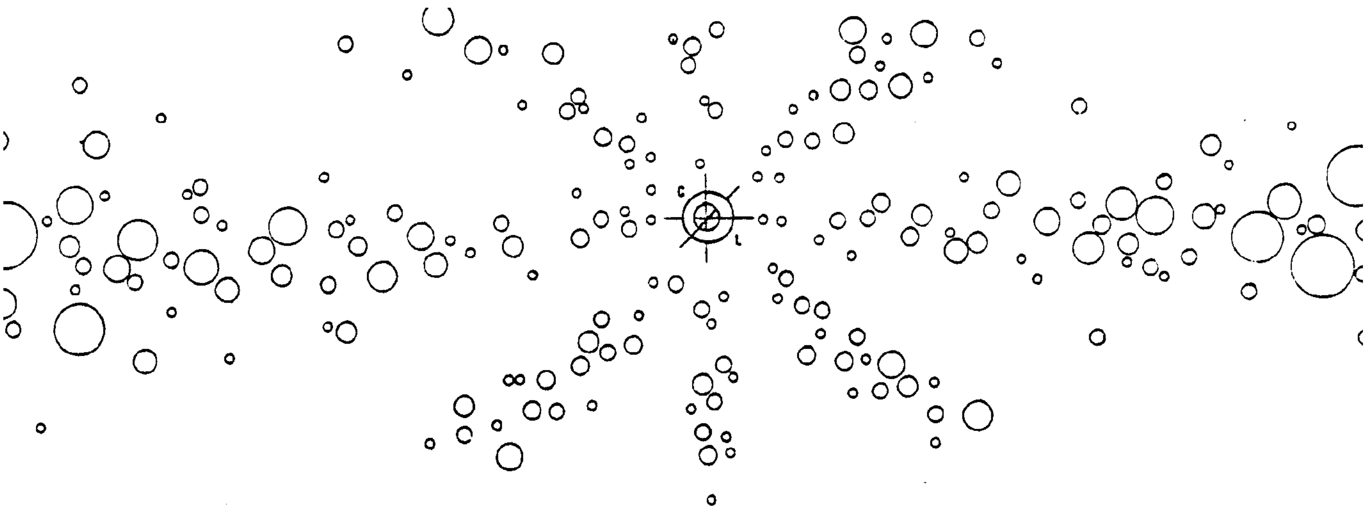


SCOPE/HUSTLER

Reference Manual

MICHIGAN STATE UNIVERSITY COMPUTER LABORATORY



KR[illegible]
SWAKERY
YUNCTIONS (YU[illegible])

SCOPE/HUSTLER
Reference Manual

Sorry – the print quality of this manual is poor, even though it is an original.

Mark Riordan
21 Oct 2003

Computer Laboratory

Michigan State University

REVISION RECORD

Revision	Description
A	This revision reflects the removal of the 3600 and the user changes introduced by LSD 32 and 33. It also corrects numerous errors found in the original printing. Major additions: ERRS, FILES, LISTPF, LISTTY, PNPURGE, PAPERT, Priority Scheme, Parity Error Procedures, and Special Problems (for magnetic tapes). (5-74)
B	In this revision, the AUTHORF utility is documented. Appendix B is removed and its contents are incorporated into Chapter 2. Chapter 9 is removed; the remote batch system is documented in User's Guide Supplement: Remote Batch Terminal Operation. (8-77)
C	This revision updates Chapter 5 on permanent files, to include the utilities PFLIST, PFDUMP and PFLOAD, deleting obsolete material. (1-78)
D	This revision updates Chapter 6, Magnetic Tapes, to include 9-track tape usage. Appendices A, C, E, and I are updated. Chapter 10 (software products) is deleted; software products are discussed in the Facilities and Policies Handbook. Appendices C, G, and H are deleted. (7-78)
E	This revision reflects the ASCII character set conversion(12-78)
F	This revision corrects some errors in Chapter 2 and adds documentation of auto-exec. Chapter 7 and Appendix J have

Additional copies of this publication
may be obtained from the User
Information Center of the MSU
Computer Laboratory.

Address comments concerning
this publication to:

User Information Center
Computer Laboratory
Michigan State University
East Lansing, Michigan 48824

or use the comment sheet at
the back of this publication.

REVISION RECORD (Cont'd)

Revision	Description
	been rewritten to include documentation of all control
	statements. Portions of Chapter 8 are removed. (5-79)
G	Chapter 4 on file structure rewritten. Appendix J updated
	and reissued. Chapter 7 on control statements updated to
	include FTN 5 and the F45 conversion aid. (1-80)
H	Chapter 1 on System Description and Chapter 3 on Job
	Structure are rewritten. (3-81)

Preface

The *SCOPE/HUSTLER Reference Manual* is the primary source of reference for MSU's SCOPE/HUSTLER operating system. References to Control Data publications or other Computer Laboratory publications are made where appropriate throughout the manual.

This manual will be updated as needed to ensure that it remains both current and accurate. Copies of this and all other Computer Laboratory publications may be obtained in the User Information Center, Room 313 Computer Center.

The following people, all past or present staff members of the User Information Center, were instrumental in producing the *SCOPE/HUSTLER Reference Manual*: Deborah Alpert, Elaine Currie, Susan Gossman, Steve Groll, Bobb Howard, Steve Huyser, Jim Lukey, Karen Overton, Tory Sawyer, and Dianne Smock. Special acknowledgements go to the Systems Development staff of the Computer Laboratory, who helped immensely by reviewing the text, testing examples, and answering technical questions.

Table of Contents

Page

INTRODUCTION

APPENDICES

Supplements to the *SCOPE/HUSTLER Reference Manual*

Introduction

SCOPE/HUSTLER is a locally developed extension of the CDC SCOPE and NOS/BE Operating System. The principal difference is HUSTLER's ability to process both batch and interactive jobs, using a unified scheduling algorithm. SCOPE/HUSTLER has been modified to accommodate most CDC SCOPE compilers and other major products.

The *SCOPE/HUSTLER Reference Manual* was written with one major objective: to provide reference documentation of all SCOPE/HUSTLER features accessible through batch control cards.

The *SCOPE/HUSTLER Reference Manual*, as the title implies, is not a suitable starting point for someone who has never used the system. Even experienced users may find sections difficult to read if they have no experience with that particular feature.

For the benefit of less experienced users, this document devotes several sections to explaining the terms and concepts that underlie SCOPE/HUSTLER procedures.

Interactive system users should note that all commands described in this volume may be executed interactively except those that request magnetic tapes. The emphasis of this manual, however, is on batch usage. Commands and procedures that relate specifically to interactive use are described in the *Interactive System User's Guide*.

Reading Guide to the SCOPE/HUSTLER Reference Manual

By focusing on different types of users in different sections, we have tried to address this manual to as many users as possible—from the occasional user of a statistical package to the experienced programmer. In view of this approach, we have prepared the following Reading Guide.

Inexperienced Users

This category comprises beginning programmers and also non-programmers who want to use a specific facility such as SPSS (Statistical Package for the Social Sciences). Usually this type of user wants just enough information to be able to start running jobs.

1. You should read Section 2.1 to become familiar with the authorization system, and PN managers (users in charge of a computer account) should eventually read all of Chapter 2. You should also read Section 2.2 to learn how to change the password for your subaccount. Although not essential, most users are eager to learn how computer costs are calculated, which is explained in Section 2.7.

2. Chapter 3 explains the basic differences between batch and interactive processing, including deck structure and job submission.

3. Basic file terminology is explained in Section 4.1. Programmers should also read the introduction to file structure presented in Section 4.2.

4. To store programs or data on the computer between runs, you will normally use permanent files, which are fully described in Chapter 5. If you want to store large amounts of information but are faced with a tight budget, you will be interested in Chapter 6, a comprehensive guide to magnetic tape usage. For beginners, the most important parts are Section 6.3 (obtaining a tape), 6.5 (tape drive reservation), and 6.5.1 (the REQUEST control statement).

New Users From Other Installations

Each year, the MSU computer system gets new users who have already gained experience using a different computer—typically, an IBM system at another university. If you are this type of user, you need most of the information outlined above for the inexperienced user, but you also have enough background to appreciate an overview of how SCOPE/HUSTLER operates and the facilities it offers.

1. Chapter 1 outlines the features of the CDC mainframe computer and the SCOPE/HUSTLER operating system. It also explains basic concepts employed in the design of SCOPE/HUSTLER.

2. Section 4.3 explains logical file structure and Section 4.4 explains file manipulation using read/write requests.

3. To familiarize yourself with the range of commands provided by SCOPE/HUSTLER, browse through the control statement summary in Appendix J. References to full descriptions are included so that you may investigate further any control statements that interest you.

Experienced SCOPE/HUSTLER Users

Users who are already familiar with SCOPE/HUSTLER will be interested in this manual primarily as a reference source for commands and procedures that are not described anywhere else. The principal reference sections are Chapters 7 and 8 and the Appendices.

1. Chapter 7 devotes a separate section to each control statement available to the batch user on the MSU system. In some cases, a section consists only of an abstract and a reference to another part of this manual, or to another publication containing the full description.

2. Because magnetic tape processing has been extensively revised in SCOPE/HUSTLER, you should rely on Chapter 6 for all tape-related commands.

Notation

The following conventions are followed when describing the format of SCOPE/HUSTLER commands.

UPPER CASE	required item which must appear as shown (but may be typed in lower case)
lower case	item must be supplied by user
|	separates alternate forms
{ }	encloses alternate forms
[ ]	encloses optional forms
________	underscores default form
= = = =	underscores abbreviation

1

System Description

This chapter presents an overview of the computing system operated by the MSU Computer Laboratory. The first section deals with the hardware configuration, while the second and third sections outline the facilities of the SCOPE/HUSTLER operating system. The final sections discuss the interrelations between hardware and software on the MSU system.

1.1 Hardware: The CDC Cyber 750 Computer

The Computer Laboratory operates a Control Data Corporation Cyber 170 series, model 750[1] computer. The Cyber 750, housed on the second floor of the Computer Center (the central site), consists of one Central Processing Unit (CPU) and 17 Peripheral Processing Units (PPU), 196608 (decimal) words of central memory (CM), and 500000 words of Extended Core Storage (ECS). The Cyber 750, the heart of the academic computing system, is often referred to as the "mainframe."

Several other, smaller, computers are connected by data channels to the Cyber 750, and perform special functions.

- The "Front-end" (FREND) computer handles the communication chores between the mainframe and a wide variety of interactive terminals, minicomputers and input/output devices at both remote and central site locations. More than 100 terminals or minicomputers can be connected to the mainframe at one time. The Front-end computer is a Perkin-Elmer 7/32 minicomputer.

- A PDP-11 minicomputer acts as a communicator between the MSU computing facilities and those at the University of Michigan, Wayne State University, and Western Michigan University, through the Merit Computer Network. The network allows users having access to the facilities at each university to use the resources at the other sites. (See Section 1.1.7 for further discussion of the Merit Network.)

- A Hewlett-Packard (H-P) 2000 ACCESS system is available as an ongoing experiment regarding the role of minicomputers in instructional computing. The H-P 2000 is a 32-port interactive system, which has an extensive library of software written in the BASIC programming language. (Note: jobs may be submitted from the H-P 2000 to the mainframe for execution.)

The mainframe is connected by data channels to a variety of input/output and secondary storage devices, including card readers and printers (see Section 1.1.4), magnetic tape drives (see Section 1.1.3), disk storage units (see Section 1.1.2), and the controllers associated with these devices.

[1] Throughout Computer Laboratory documentation, this name is shortened to "Cyber 750."

Figure 1-1, a connection diagram, shows the complete hardware configuration of MSU's computer system. The general flow of information through the system may be summarized as follows:

Programs and data enter the system through card readers or terminals, linked by direct cable or telephone lines from central site or remote locations. Information from these primary input devices is transferred by the peripheral processors (PPs) through the data channels to one of the disk storage units. When a job is selected for execution, the peripheral processors transfer requested programs and data from disk or tape into central memory, where they are accessible to the central processor (CPU). Note that the CPU communicates only with central memory (CM), extended core storage (ECS) and the PPs. Output generated by the job follows a similar path in the opposite direction.

1.1.1 The Central Computer

The mainframe consists of one central processor, 17 peripheral processors, 24 data channels, 196,632 words of central memory, and 500,000 words of extended core storage.

The Central Processor (CPU)

The primary function of the central processor is to perform the computational part of user programs. The CPU has a built-in set of instructions which are oriented toward floating-point arithmetic, address in memory calculation, and arithmetic decision-making.

Although the central processor is the main component of the system from your point of view, it requires "peripheral processors" to communicate with external devices. The CPU does not have any input/output instructions and communicates only with central memory, extended core storage and the peripheral processors. To get information into and out of central memory, from and to external devices, a CPU program must submit a request to a PP program, using central memory communication areas defined by the operating system.

For a more detailed look at the workings of the CPU, see Section 1.4.1.

Peripheral Processors (PPU or PP)

The peripheral processors control the flow of information between input/output devices and the central processing unit. They are identical computers that execute stored programs independently of each other and of the CPU; each has its own memory.

The PPUs are less complex and slower than the CPU, and perform various utility functions and input/output activities so that the CPU is free to work strictly on calculations. User-written programs cannot access PPUs directly.

Central Memory (CM)

The central memory in the Cyber 750 is a metal oxide semiconductor (MOS) memory.

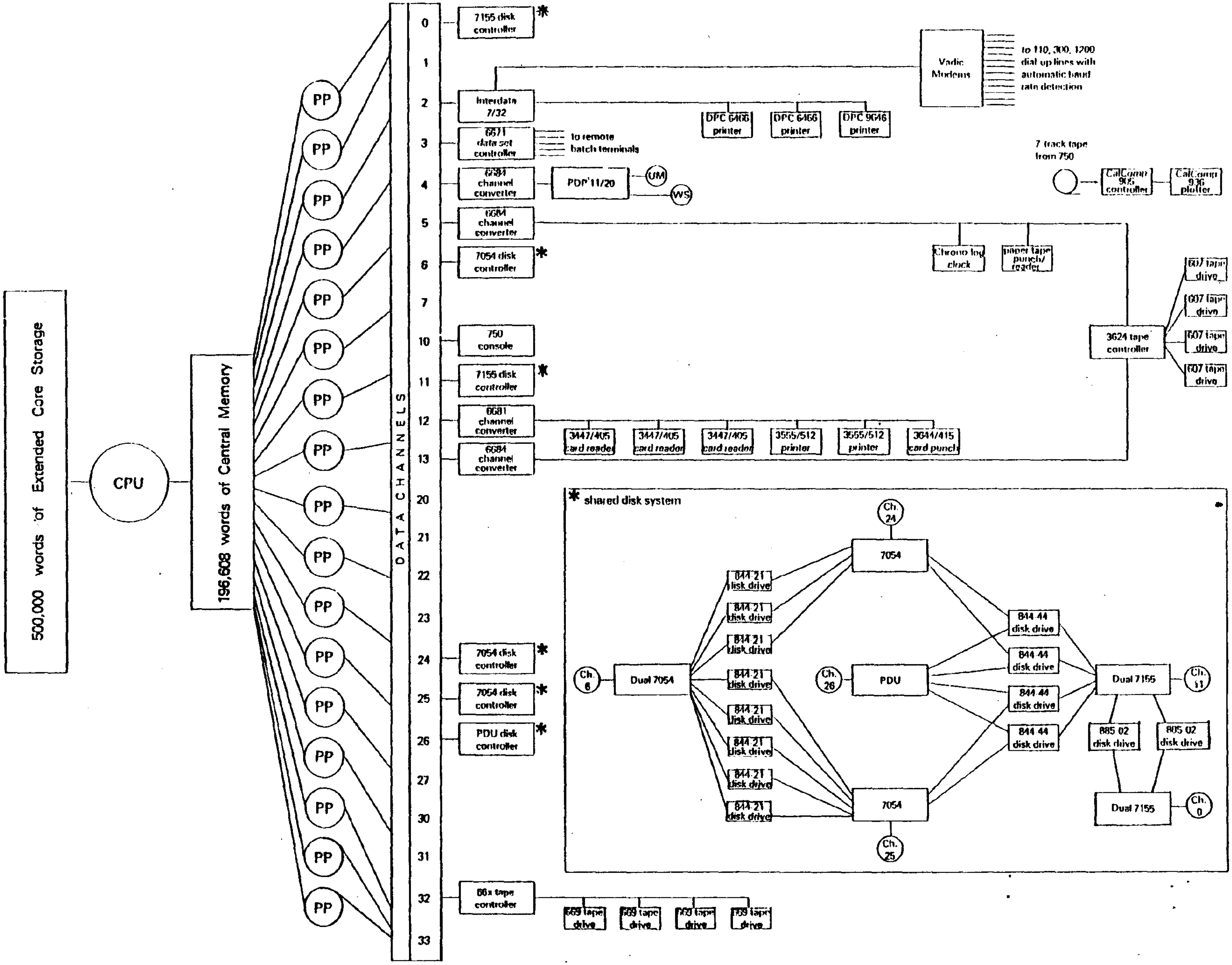

500,000 words of Extended Core Storage
CPU
196,608 words of Central Memory
PP
DATA CHANNELS
0
1
2
3
4
5
6
7
10
11
12
13
20
21
22
23
24
25
26
27
30
31
32
33
7155 disk controller
Interdata 7/32
6671 data set controller
to remote batch terminals
6684 channel converter
PDP 11/20
UM
WS
6684 channel converter
7054 disk controller
750 console
7155 disk controller
6681 channel converter
6684 channel converter
7054 disk controller
7054 disk controller
PDU disk controller
66x tape controller
669 tape drive
669 tape drive
669 tape drive
669 tape drive
Vadic Modems
to 110, 300, 1200 dial up lines with automatic baud rate detection
DPC 6466 printer
DPC 6466 printer
DPC 9646 printer
7 track tape from 750
CalComp 905 controller
CalComp 936 plotter
Chrono log clock
paper tape punch/ reader
3624 tape controller
607 tape drive
607 tape drive
607 tape drive
607 tape drive
3447/405 card reader
3447/405 card reader
3447/405 card reader
3555/512 printer
3555/512 printer
3644/415 card punch
* shared disk system
Ch. 24
7054
844 21 disk drive
844 21 disk drive
844 21 disk drive
844 21 disk drive
844 21 disk drive
844 21 disk drive
844 21 disk drive
844 21 disk drive
Ch. 6
Dual 7054
Ch. 26
PDU
844 44 disk drive
844 44 disk drive
844 44 disk drive
844 44 disk drive
Dual 7155
Ch. 11
885 02 disk drive
885 02 disk drive
Dual 7155
Ch. 0
7054
Ch. 25

Figure 1-1

The functions of central memory are:

- to hold the programs being executed by the CPU,
- to store the values that a program is working with, and
- to hold a number of tables and buffers used by the operating system.

Information is represented by binary digits (bits), grouped into "words" of 60 bits each. Each word typically contains data encoded in:

- numeric form-including floating point and integer data, or
- character form-including "display code" or "ASCII."

Display code characters require 6 bits of a 60 bit word whereas ASCII characters require 12 bits.

The CPU must be able to locate each of the words in memory. To accomplish this, each word is assigned an **address**. The addresses assigned to words in memory are used by the CPU to refer to values that are stored in those words. In order to allow the CPU to operate at top speed, the time needed to store or retrieve the contents of any word of memory must be very short. This time is called "access time"; the access time for the Cyber 750's memory is 400 billionths of a second (nanosecond).

Extended Core Storage (ECS)

Extended core storage (ECS) is designed for high speed transfer of data to and from central memory. Although slower than central memory, ECS transfers data about 100 times faster than disk storage. Only the SCOPE/HUSTLER operating system has access to ECS through central memory; this storage medium is not directly available for users' programs.

Operator's Console

A human operator monitors the activities of the Cyber 750 computer system via the operator's console, which consists of a keyboard and a cathode ray tube (CRT) screen. While the console's normal function is operator-system communication, it is also used for system debugging and hardware testing.

1.1.2 Disk Storage

Central memory and ECS are used by jobs being executed by the CPU at any given time. Because this high speed storage is so expensive, auxiliary storage media are necessary. Magnetic disks are the principal storage medium of the SCOPE/HUSTLER operating system. Disk storage is used for both temporary and long-term storage. The disk system for the Cyber 750 currently provides over two billion characters of storage space; about 100 million characters are reserved for use by the operating system and the rest are available for user jobs.

Disk storage is organized in the following manner. Each disk is divided into tracks; each track being divided into sectors of 640 characters (64 central memory words). A sector is referred to as a **physical record unit** (PRU) because it is the smallest unit of data that can be transferred to or from the disk.

Physical file structure is discussed in more detail in Section 4.2.

1.1.3 Magnetic Tape Storage

Magnetic tape is used as a backup storage medium, for data transferred from one computer to another, and for storage of large files and data bases.

Magnetic tapes may be purchased at the Service Window in Room 208 Computer Center. Tapes may be obtained with a Tape Service Request Form. There are two types of storage: temporary and permanent. All tapes used with the SCOPE/HUSTLER system must be stored in the Computer Laboratory's tape library. This policy reduces user handling and tape malfunctions. More information on the Computer Laboratory's policy regarding tapes can be found in Chapter 7 of the *Facilities and Policies Handbook.*

The Cyber 750 has four 7-track tape drives and four 9-track tape drives. Both use standard, one-half inch magnetic tape. The tape drives read and write data at various densities (in cpi, or characters per inch) and speeds, as follows:

Type of Tape Drive	Data Densities	Speed of Tape drive	Equivalent to
	(6-bit characters)		
7-track	200 cpi*	150 in/sec	30,000 char/sec
	556 cpi	150 in/sec	83,400 char/sec
	800 cpi	150 in/sec	120,000 char/sec
	(8-bit characters)		
9-track	800 cpi	200 in/sec	160,000 char/sec
	1600 cpi	200 in/sec	320,000 char/sec

Magnetic tapes are considered to be a slow storage medium because human intervention is required in order to mount a tape on a tape drive, and because access to the data is on a sequential basis only. The use of magnetic tapes is described in detail in Chapter 6.

*Data should not be written at 200 cpi.

1.1.4 Central Site INPUT/OUTPUT Equipment

The following input/output devices are located at the central site Computer Center.

Card Readers

The central site is equipped with three card readers, each of which has a maximum reading rate of 1200 cards per minute. Each card reader is connected by direct cable, and operates under the control of the SCOPE/HUSTLER operating system. The readers are available on a self-serve basis during production hours.

Line Printers

Several line printers are located at the central site. They are of two types: those that print Display Code, and those that print either ASCII Fancy (upper/lower case ASCII) or Display Code. (For more information about character codes see Section 4.5.) Each printer has a maximum line width of 136 characters; you can select either six or eight lines per inch.

The line printers are connected by direct cable, and operate under the control of the SCOPE/HUSTLER operating system. All but one of these printers are monitored by the Input/Output (I/O) Room staff. Output is separated and filed in bins by staff members in Room 208. The exception is a self-service printer in Room 208, which is accessible as source "A" (see Section 7.4.1 and Appendix E.).

Card Punch

Output can be punched in binary or display code format on the card punch located in the Input/Output Room. The card punch, run by I/O Room staff, operates at a rate of 250 cards per minute. When punched output is finished, it is filed in card files in Room 208.

Plotter

If you're interested in graphics output you can make use of the CalComp incremental pen plotter at the central site. You have a choice of four ink colors, two paper widths, and several paper types and pen sizes. Up to three pens may be used for one plot. Arrangements for plotting output can be made at the Service Window in Room 208. Refer to the *Plotting and Graphics Reference Manual* for more information.

Keypunch Machines

Keypunch machines are available for use in Room 210 Computer Center on a first come, first served basis. These machines all punch IBM 026 code, which is the standard code accepted by the central site card readers. Cards can be purchased in small quantities in vending machines in Room 208 Computer Center; boxed cards can be purchased in the Main Office (220 Computer Center).

Lister

The lister, located in Room 208, may be used to list your card deck for proofreading purposes before reading the deck into the computer system. This service, offered free of charge, saves time and card handling.

Status Display Monitor

The status display monitor is a video monitor that displays the status of central site batch jobs in the system. Located in the User Work Room (Room 212 Computer Center) the monitor identifies jobs by their sequence numbers. After displaying all stored information once, the status is updated and the display is run again; this occurs continually during production time. The monitor is connected by direct cable, and operates under the control of the SCOPE/HUSTLER operating system.

Graphics Laboratory

The Graphics Laboratory, located in Room 508 Computer Center, is available on a first come, first served basis. A digitizer and Tektronix 4012 graphics terminal may be used by students, faculty and staff. The key may be obtained at the Service Window in Room 208. The digitizer and Tektronix terminal are both connected by direct cable and operate under the control of the SCOPE/HUSTLER operating system.

1.1.5 Remote Batch Terminals

Several low-speed remote batch stations are available for use with the Cyber 750. The remote batch stations are located on campus and around the state. Each station consists of the following equipment: a cathode ray tube (CRT) terminal, a card reader, a line printer and a controller/emulator. These terminals are connected to the central site via hardwired lines or dial-up modems. Each remote station is assigned a site identifier (the second character of the job sequence number).

1.1.6 Interactive Terminals

Interactive (conversational) access to the Cyber 750 is accomplished by low-speed interactive terminals. Terminals are machines used for communication with the mainframe; sending instructions by means of a keyboard and receiving the computer's response on paper or on a cathode ray tube (CRT) screen.

Public terminal areas are located in Room 208 Computer Center, the Undergraduate Library, Conrad Library, Brody Hall, and various other sites on campus. In addition, many departments and individuals own or lease terminals for use with the mainframe. Many of the public terminals are "hardwired", which means that the terminal is physically linked and automatically connected to the computer when the terminal is turned on, or when a special "log-in" switch on the terminal is pressed. If a terminal is not hardwired, it requires the use of a telephone and an acoustic coupler (modem) to link with the mainframe. The appropriate telephone number is dialed and the handset is placed into the acoustic coupler; data is transmitted over the phone lines.

The transmission rates of data to and from the terminals vary. The current supported transmission (baud) rates are 110 (10 characters per second), 300 (30 characters per second), and 1200 (120 characters per second).

Communication chores for the SCOPE/HUSTLER interactive system are handled by the Front-end system (FREND). The Front-end computer is connected to the mainframe via a high-speed channel interface. Phone lines in the Front-end system enable interaction between the Cyber 750 and a wide variety of terminals and minicomputers. The Front-end offers keyboard editing functions as well as a set of device commands that modify certain terminal characteristics and input/output functions. See Chapter 8 of the *Interactive User's Guide* for more information about the Front-end.

1.1.7 Merit Network

The Merit Computer Network connects the MSU computing system to those at Wayne State University, the University of Michigan and Western Michigan University. The network allows all users with current access to the facilities at each university to use the resources at any of the other sites, if appropriate authorization is acquired.

The computer at each center (host) is interfaced to a PDP-11 minicomputer, called a Communications Computer (CC), which preprocesses the incoming and outgoing information to make it compatible with the host. The CCs transmit and receive information over standard telephone lines.

The Merit Network provides direct interactive access through Hermes, Merit's network-to-terminal interface. Dialing directly into Hermes provides access to any Merit host computer without going through the local system. If you are not near one of the four network nodes, Hermes can be accessed by a long-distance telephone call or through Telenet. Besides providing network access, Hermes offers keyboard editing functions as well as a set of device commands that modify certain terminal characteristics and input/output functions.

Telenet is an international telecommunication network linked with Merit. It allows interactive use of the Merit host computers from anywhere in the United States and a score of foreign countries. Currently the link is inbound only, from Telenet into Merit.

1.2 Software: The SCOPE/HUSTLER Operating System

SCOPE/HUSTLER consists of a group of programs, subprograms, and data tables that form the operating system for the MSU mainframe computer. It was derived from the standard CDC SCOPE and NOS/BE operating systems, with extensive modifications made by the Computer Laboratory Systems Development group. One of those modifications was the creation of HUSTLER, a scheduling mechanism that features the integration of batch and interactive processing.

SCOPE/HUSTLER has several important functions:

1. Allowing access to the computer
2. Scheduling execution of batch and interactive jobs
3. Organizing information within the computer
4. Long-term storage and protection of information
5. Accounting and authorization
6. Providing utilities to aid users in processing jobs

In short, its functions are to ensure efficient use of the computer's capabilities and to provide support services for users. The following sections take a closer look at the specific functions.

1.2.1 Access: Interactive and Batch Processing

Access to the SCOPE/HUSTLER system is gained through interactive and batch processing.

In batch processing, you prepare a complete job and submit all of your instructions to the computer at one time. Batch processing normally relies on the use of card decks which are read into the computer via a card reader; in addition, batch jobs can be submitted from an interactive terminal (see Section 3.6.2 of this manual and Section 7.2 of the *Interactive User's Guide*).

In interactive processing, you perform a task step-by-step, issuing instructions individually and receiving a response from the computer after every piece of information is processed. Interactive processing involves the use of computer terminals.

The interactive system receives and prepares commands for processing by SCOPE/HUSTLER and relays output back to your terminal. Except for magnetic tape requests, all commands recognized in batch mode can be entered under the interactive system. Since in interactive mode you may create and submit files for processing as batch jobs, this limitation can be overcome by creating a batch job that will request a magnetic tape and copy its contents to a disk file, which can then be accessed from a terminal.

Because of the special nature of interactive processing, there are a number of commands which have no equivalent in batch use; these are referred to as "interactive commands" as opposed to "SCOPE/HUSTLER commands." See Chapter 7 of this manual and Chapter 2 of the *Interactive User's Guide* for further information.

1.2.2 Job Scheduling

The job scheduling structure of SCOPE/HUSTLER was developed to integrate batch and interactive processing. It is based on the following components:

1. a scheduling structure, known as the job pool, designed to hold both batch and interactive jobs;

2. a swapping routine called by the scheduler to transfer jobs back and forth between central memory and disk storage or ECS; and

3. a unified scheduler which repeatedly examines all jobs in the pool and selects the "best" combination for execution at the control points (areas in central memory).

Four times or more each second, the scheduler scans the job pool and evaluates a scheduling formula for each job executing or waiting to execute. This formula is designed to satisfy the following needs:

1. to minimize response time for interactive users,
2. to ensure equal machine utilization for all users, and
3. to maximize utilization of central memory.

Timesharing is an important element of interactive processing. Interactive jobs are initially given a higher priority than batch jobs, to minimize response time. The scheduler tracks the length of time a batch job remains in the system to ensure that it is not unnecessarily delayed.

The concepts of "job pool," "control points," and other terms related to job processing are explained in Section 1.4.

1.2.3 Organization of Information Within The Computer: Files

A file is a set of logically related information which is stored on an external device, such as a magnetic tape or disk. One of the distinctive features of SCOPE/HUSTLER is that all information is handled in terms of files. In other words, each job, each program, each collection of data, and each set of output is either a file or part of a file.

The comprehensive use of files in SCOPE/HUSTLER is designed to minimize device-dependent procedures. In general, you can read or write a file, or copy data from one file to another, without regard to where the files are stored. For detailed information on files, see Chapter 4.

The Role of Files in Batch Processing

The role of files in batch processing within the SCOPE/HUSTLER system can best be seen by tracing the progress of a batch job through the system. The first step in preparing a job for batch processing is to organize a card deck into a **file** composed of one or more **sections**, where each section is a sub-deck terminated by a special end-of-section card. The first section, called the control section, consists entirely of control statements, i.e., statements containing commands to SCOPE/HUSTLER (see Chapter 3). Subsequent sections, if any, contain data for the programs called in the control section.

By reading in the card deck, the system transfers the information to disk storage, assigns it a file name (the job sequence number), and designates it as an **input file**. The system also assigns the file a priority computed from the job cost and rate group parameters of the job card (one of the initial control statements). Ordered by priority, the input files form an **input queue**. When a slot becomes available, the system "removes" a file from the front of the input queue and assigns it an entry in a scheduling table known as the **pool**. The pool is comprised of all jobs eligible for execution at a **control point** (see Section 1.4.3).

When the job begins execution, the job file is changed from type input to type **local**, indicating that it "belongs" to a job. At the same time, the file name is changed from the job sequence number to INPUT. Then the first section of INPUT is read into an area of memory used by SCOPE/HUSTLER as its source of control statements, and the file is left positioned at the start of the first data section. Reading from file INPUT, therefore, is equivalent to reading data cards from the job deck. The system also creates a local file named OUTPUT to collect system-generated program listings, diagnostics, maps and dumps, and any output you choose to write on it. Unless you request some other processing, file OUTPUT will be routed to a printer and all local files will be eliminated from the system at the end of the job.

A job consists of one or more programs, sequenced in order of desired execution. When these programs are executed, they generally establish local files, used for the duration of the job's execution, either to hold output or to access programs and data saved by previous jobs. Magnetic tape files and disk-resident permanent files must be explicitly created and requested with a REQUEST or an ATTACH command before they are used. Local disk files are implicitly (automatically) created when specified by assigning a name (often referred to as an "lfn," which is short for local file name) in a control statement option or when referenced in your program. In addition, many system utilities create local files without any specification from you.

For the sake of illustration, suppose your job consists of the following card deck.

```
PNC
Job card
password
FTN5.
LGO.
CATALOG,TAPE1,MYPERMANENTFILE.
7/8/9
      PROGRAM SAMPLE
      .
      .
      .
7/8/9
data
6/7/8/9
```

Aside from the initial identification cards, each control statement may be considered a request to load and execute a program from the file, part of a file library, indicated by the control statement flagword (see Section 1.3.1 for discussion of control statements). In this example, the "FTN5." card refers to the FORTRAN Extended version 5 compiler stored on the system library. The compiler uses several files: by default, it reads the second section of INPUT (the source cards for program SAMPLE); it writes a source language (the code you submit) listing of the program on OUTPUT; it writes the generated object code (code produced by the compiler) on LGO (a default file name); and it creates a number of scratch files for its own use.

The next control statement, "LGO.", requests the system to load and execute the contents of LGO, the local file just created by FTN5. SAMPLE, the program contained on LGO, uses implicitly created local files INPUT and TAPE1. The CATALOG statement changes the status of TAPE1 from a local file to a permanent file named MYPERMANENTFILE.

At the end of the job, all local files associated with the job are disposed of in the following way. If the file resides on a magnetic tape, the tape is rewound and unloaded and the local file is dropped from the system. If the file resides on a disk unit, the system first determines whether you have requested that it be printed or punched. If so, the file is changed from type local to type output, its name is changed to the job sequence number, and it is retained by the system until is has been printed or punched. To print or punch the contents of a file, you assign a disposition code, which specified how the file is to be processed after it has been released from the job. For convenience certain file names, such as OUTPUT and PUNCH, have default dispositions.

Options are available to specify how a local file is processed after it is released from your job. See the DISPOSE command in Section 7.4.1 for further discussion.

Local disk files that are not given a disposition are immediately discarded at the end of the job. To retain this type of file for use by subsequent jobs, you may CATALOG it as a permanent file (see Chapter 5).

The Role of Files in Interactive Processing

Because of the nature of interactive processing, the role of files is somewhat different than in batch processing. The concepts of local files, permanent files, and file disposition apply to interactive jobs. Although the file names INPUT and OUTPUT no longer have any special meaning, others such as PUNCH, PUNCHB and P80C, retain their status. The role of input and output files is largely replaced by **connected files.**

Interactive users communicate with executing programs using connected files, which transmit information to and from the terminal rather than store it on the disk as in batch. Once a file is connected it can be used for input, output or both; that is, there is no distinction between files connected for read or write operations. (Note that a READ request on a connected file causes the program to pause until input has been received from the terminal keyboard. A WRITE request on a connected file causes the data to be immediately displayed or printed at the terminal.) You may connect any temporary file at any time. All file positioning commands are ignored when made on connected files. See Chapter 5 of the *Interactive System User's Guide* for further information.

An interactive job may be assigned a special EDITOR work file named EWFILE, which can be used to construct and edit files.

File structure, permanent files, and magnetic tape files are discussed at length in Chapters 4, 5, and 6 of the this manual. EDITOR work files and connected files are covered in Chapters 3 and 5 of the *Interactive System User's Guide.*

1.2.4 Protection and Storage of Files

As already mentioned, the SCOPE/HUSTLER system is based on files and file storage. A file kept on disk storage in the computer system is a permanent file. Permanent files make it easy to share information among several jobs.

There are five types of permanent file access: access (turnkey), read, extend, modify, and control. Each can be protected with a separate password. The location and identification of permanent files are maintained by the system in special, disk-resident tables. These tables ensure that a permanent file is not destroyed when a job using it ends. See Chapter 5 for more information on permanent files.

MSU has invested considerable effort in devising a back-up tape storage system for permanent file protection. To improve security of tape files the tape visual reel number and your problem number are automatically checked when the operator attempts to assign a labeled tape. Rewriting a labeled tape is restricted to the PN owner of the tape unless the owner has specified no protections. See Chapter 6 for more information on magnetic tapes.

1.2.5 Accounting and Authorization

An important factor in authorization on the SCOPE/HUSTLER system is the Authorization File. Several identifiers are required to execute a job:

1. a **problem number** (main account),

2. a **User ID** (sub-account), and

3. a **password** (a protection for the ID).

The problem number (PN), its IDs, passwords, PN and ID dollar balances, and problem number limits are recorded in the Authorization File. Before allowing a batch job to execute, SCOPE/HUSTLER checks the Authorization File to verify that:

1. the PN is authorized,
2. the specified ID is authorized for the PN,
3. the specified password is correct (if required),
4. the PN and ID dollar balances are positive,
5. the requested job limits are within the authorized PN limits, and the requested job cost limit is within the current ID and PN dollar balances.

If a job fails any of these tests, it is aborted immediately. In interactive mode you are given three chances to correctly enter your PN, ID and password. During execution, authorization data is used to determine whether certain types of control statements are authorized and whether certain resource requests are within the PN limits.

Use of various system resources, such as central memory space, is controlled by a set of values known as **problem number limits**. Overall usage is controlled by a **dollar balance**, which is kept for the problem number and for each of its IDs. The cost of each job you run is decremented from both your problem number and ID dollar balances.

SCOPE/HUSTLER monitors the computer services (CPU time, PP time, central memory usage, etc.) and supplies used by each job and records these statistics in the **system dayfile** (see Section 2.7.3). A partial summary of the job cost is printed at the end of each job in the **job dayfile** (see Section 3.7.6). To provide greater flexibility and responsibility for your accounts, you are classified as either problem number managers or as subaccounts of a manager's problem number. Users who receive their authorization directly from the Computer Laboratory are designated as **problem number managers**. A problem number manager can, in turn, authorize other individuals by assigning each a user ID, a password, and a portion of the problem number dollar balance. A problem number manager can control the dollar balances allotted to each of the user IDs as well as many of the problem number limits. For more detailed information, see Chapter 2.

1.3 Features of SCOPE/HUSTLER

The basic function of the SCOPE/HUSTLER system is to ensure efficient use of the computer's capabilities and to provide support services for users. The following sections outline some of the facilities of SCOPE/HUSTLER.

1.3.1 Control Statements

Control statements are commands to execute a program in the operating system, user library, or a local file.

In batch mode, control statements appear only in the first section of the job. This section consists of identification and authorization cards: sequence card, problem number card, job card and password card (if required; see Chapter 2). In interactive mode, the identification and authorization statements are combined on one line. Other control statements follow and specify how the job is to be processed.

The syntax of each control statement consists of a flagword, followed (optionally) by a parameter list, followed by a control statement terminator. When parameters are specified, the flagword is separated from the first parameter by a delimiter; remaining parameters are also separated from one another by delimiters. Control statements are discussed in detail in Chapters 3 and 7.

1.3.2 Program Compilation

A variety of compilers and assemblers are available. A compiler is a language processor which translates programs, written in a particular programming language, to machine language. An assembler is a language processor for an assembly language, a computing language in which each machine language instruction is represented in symbolic notation.

The major compilers and assemblers available on the SCOPE/HUSTLER system are:

BASIC	BASIC language compiler (see CDC *BASIC v3 Reference Manual*).
COBOL 4	COBOL language compiler (see CDC *COBOL Reference Manual*).
COMPASS	Control Data Machine language assembler (see CDC *COMPASS v3 Reference Manual*).
FTN 4	FORTRAN IV Extended language compiler (see CDC *FORTRAN Extended v4 Reference Manual*).
FTN 5	FORTRAN 5 Extended language compiler (see CDC *FORTRAN Version 5 Reference Manual*).
PASCAL	PASCAL language compiler (see *PASCAL User Manual*).

1.3.3 Applications Software

A variety of language processors, utility programs, and application packages are available to users of the computer system. Most of the major software products are supplied and supported by Control Data Corporation. See Appendix D for a list of supported software.

In addition to the CDC products, the Computer Laboratory has obtained many programs and packages through exchange agreements with other computer installations. These programs reside in either the HUSTLER Auxiliary Library or the Unsupported Library, UNSUP.

The HUSTLER Auxiliary Library is a working library. Programs are stored on disk files or tapes, and are retrieved by means of the system utility, HAL. Thus, programs on the library are accessed with this general control statement:

```
HAL,programname,parameters.
```

To find out if a particular program is on the HUSTLER Auxiliary library, or to obtain a description of that program, use the command

```
HELP,programname.
```

or refer to Chapter 7 of this manual.

A variety of programs reside in the UNSUP Library. In general, these are programs written for a specific purpose which may be useful, in whole or part, to other users. You may access a list of the programs on the UNSUP Library by using HELP in the form,

```
HELP,L*UNSUP,UNSUP.
```

Descriptions of individual UNSUP entries are available via 'HELP,L*UNSUP,entryname.'

1.3.4
Debugging Aids

Debugging aids are useful if you are interested in saving time and money. Debugging aids can be divided into the following categories:

Interactive Debugging Facilities
Compilation Aids
System Error Messages
Dumps
Loader Error Detection

For more information on available debugging aids, see Chapter 6 of the *Interactive System User's Guide* and Chapter 7 of this manual.

1.3.5
System and User Libraries

SCOPE/HUSTLER provides several system libraries, containing operating system routines, utilities and compilers. MSU also maintains several libraries of frequently used programs.

A user library is a collection of program and data files together with an index to enable quick access to any of them. A variety of utilities are incorporated in the SCOPE/HUSTLER system to enable you to create, update, and use your own library of programs or data, stored in either source or binary form. Some of the utilities are:

UPDATE designed especially for large libraries of source programs. Modifications can be made on a card-by-card or a deck-by-deck basis. See the CDC *UPDATE Reference Manual.* (CDC publication number 60449900).

HAL
used to build, maintain, and use libraries having the same retrieval features as the HUSTLER Auxiliary Library. It has entry-by-entry update capabilities. See the *HAL Reference Manual*.

EDITOR
designed for the interactive system but can also be used in batch mode. Its chief attraction is intra-line editing in addition to line-by-line editing. It cannot handlė binary files. See Chapter 3 of the *Interactive System User's Guide* for interactive usage and Section 7.8.1 of this manual for batch usage.

LIBEDIT
used to build and maintain collections of relocatable and absolute binary programs and subroutines for use by the Cyber Loader. It has deck-by-deck update capabilities. See the CDC *Cyber Loader Reference Manual* and the User's Guide Supplement: *LIBEDIT—Cyber Loader Libraries*.

CYBER
LOADER
places programs into memory so that they are ready for execution. Loader input is obtained from local files and libraries. Upon completion of the program, execution of the program can be requested. Loading also performs services such as generation of a load map, presetting of unused storage to a specified value and generation of overlays or segments. See Section 3.7.3, the *User's Guide Supplement: FORTRAN Extended Library Routine* and the *CDC FORTRAN Extended Reference Manual* for further discussion.

More information on library creation facilities is available in Chapter 7.

1.4
Program Communication with SCOPE/HUSTLER

Many elements are interdependent in the working of a computer system. The information included in this section will facilitate understanding of the mainframe computer's inner workings and communication with the SCOPE/HUSTLER system.

1.4.1.
Hardware/Software Interrelations

To understand the overall design of the SCOPE/HUSTLER system, one must bear in mind the unique configuration of the Cyber 750. In summary, the central processor has very high-speed calculating abilities but lacks the means for getting information into (or out of) memory from (or to) an outside source. All input and output is handled by the peripheral processors, whose small memory word-size and simple calculating abilities are tailored to these relatively low-speed tasks.

Temporary storage devices within the CPU are called registers. The central processor has a set of 24 operating registers:

a. eight 60-bit operand registers (X0 through X7), in which most of the calculation is done;

b. eight 18-bit address registers (A0 through A7), which control the transfer of data between central memory and the operand registers; and

c. eight 18-bit increment registers (B0 through B7), which are used primarily for program indexing.

In addition, the CPU has several special purpose registers:

- The P (program address) register contains the address of the next instruction word to be executed.
- The RA and FL registers hold the *reference address* and the *field length* (see Section 1.4.3). Each job occupies a contiguous block of words in central memory. References to all addresses within each block are made in relation to the reference address (RA), the first address in the block. The length of the block is the field length (FL) of the job. At any given instant, the central processor is confined to the block of central memory starting at the address contained in RA and extending the number of locations indicated by FL. All memory addresses referenced by central processor programs are relative to the starting address held in RA.
- The EM (exit mode) register specifies certain classes of error conditions under which the CPU is to interrupt execution and notify the operating system.

See Section 1.4.5 for further information on registers.

The peripheral processors (PPs) are logically independent processors each running its own programs. They free the CPU of the system utility functions, e.g., input/output functions, updating status information, and so on. However, two facts are worth noting at this point:

1. External communication with the CPU is totally dependent upon the PPs. The CPU is responsible for functions dealing with ECS since the PPs have no direct access to ECS. Any PP, by executing one of its instructions under the coordination of software, can write into (and read from) any area of central memory. In a hardware sense, the CPU cannot directly affect any PP. Software programming conventions coordinate CPU and PP interaction.

2. When the computer system is deadstarted (see Section 1.5), the PPs are forced to read and are filled with programs from a system tape while the status and control register allows any PP to deadstart another PP or to "exit" from the current instruction. No PP can otherwise alter the registers of another PP, put information into its memory, or even read information out of its memory. Obviously, to avoid chaos the PPs must be programmed to work together. This is done using a system of "requests." See Section 1.4.6 for more information on requests.

The Cyber 750 is coordinated by loading one PP with a program called the Monitor. The CPU stores a program called CPUMTR in central memory resident. This CPUMTR and Monitor interface to coordinate CPU and PP interaction. The other PPs contain a resident routine for receiving and responding to requests from the PP Monitor. This PP resident routine repeatedly examines a central memory communications area for messages from the Monitor, instructing the PP to load and execute one of the PP system library routines. If that program must make a request of another PP, the request is relayed through the communications area to the Monitor. Thus, all PPs are programmed so that they will never do anything that is not approved in advance by the Monitor.

Deadstart stores a program called CPUMTR in Central Memory Resident (CMR). The CPU scheduling activity is coordinated by CPU software called the Corridor which decides which task or user control point should get the CPU next, processes all requests issued via exchange jump instructions, updates CPU time, and so on. CPUMTR and PP Monitor software interface to coordinate CPU and PP activities.

All SCOPE/HUSTLER routines are recorded on a disk-resident file. Most are called into memory only when needed, but copies of certain frequently used routines are also kept permanently in central memory, extended core storage, or a PP memory so they can be executed with minimal delay.

SCOPE/HUSTLER also builds a number of tables and buffer areas in central memory and ECS to communicate between user jobs and system routines, to communicate between the peripheral processors, and to record the allocation of disk storage and other machine resources. The portion of central memory in which this system information is held is referred to as CMR (Central Memory Resident) and cannot be accessed by users' jobs.

1.4.2 Multiprogramming

SCOPE/HUSTLER partitions central memory so that it can be allocated to several jobs at one time. At any given instant, only one job can be using the central processor, but several can be performing input/output. In fact, one job can have more than one input or output operation in progress simultaneously. The simultaneous execution of two or more jobs is known as **multiprogramming.**

To illustrate in greater detail the way in which the central and peripheral processors are shared, suppose that the program currently being executed in the CPU needs to process an image from file INPUT, which is stored on disk. To transfer this data into central memory, a CPU program stores a read request in a predefined memory location (RA+1, the second location of the job field length), which is repeatedly examined by a CPU system supervisory task (see Section 1.4.1).

When a non-zero word is encountered at this location, the CPU monitor will decide whether it can process the request itself (as in END, ABT, and others), pass it to another CPU Monitor task (as in CIO requests), or to assign the request to a peripheral processor. There are two kinds of responses: first, the CPU Monitor will acknowledge receipt of the request by zeroing the RA + 1 location and, later, the program processing the request will acknowledge completion of the operation by setting a flag in an appropriate location in the program.

If the request does not specify **auto-recall**, the program may continue to execute and initiate other operations. In most cases, however, the request will specify auto-recall, and the CPU Monitor will reassign the CPU to another job while the input/output is performed. Returning to our example, during the time it takes the PPs to process the read request and transfer a block of data from the disk into central memory, other jobs will be able to execute on the order of 50,000 to 100,000 CPU instructions. Upon completion of the read operation, the CPU Monitor will be notified that the job is again ready to use the CPU.

Although the preceding paragraphs describe what actually happens when a line of information is ready by the computer, the RA + 1 request, the auto-recall option, and the system acknowledgments are transparent to you except at the assembly language level.

1.4.3 A Closer Look At The Scheduling Process

SCOPE/HUSTLER permits several jobs to occupy central memory at one time and to share the central and peripheral processors by multiprogramming. To keep track of these jobs, the system assigns each to a **control point**.

Two blocks of central memory are associated with each control point: the **control point area** and the **job field length**. The control point area is a 200_8 word block which holds information used to control execution of the job. This information includes:

1. the exchange package, in which the contents of the CPU registers are saved when the CPU is released and from which the registers are restored when the CPU is reassigned;
2. a control statement buffer, containing system commands to be executed, plus a pointer to the current command;
3. job identifiers, such as the job sequence number, problem number, and ID;
4. job limits obtained from the job card and the Authorization File, and statistics on accumulated machine usage, such as CPU time, PP time and interactive connect time; and
5. flags indicating the selection of variŏus processing options.

The control point area is located in a portion of central memory which is inaccessible to you, but several utilities make it possible to retrieve some of the information contained there.

The job field length is a variable-length block of memory into which user and system programs are loaded as they are called and executed by the job. The job field lengths associated with the control points are contiguous and arranged in order of control point number. To prevent jobs from interfering with one another, each job can only reference the memory locations within its current field length. Any attempt to reference a location beyond its boundaries will normally cause the job to abort.

Central memory used by a particular job may be managed by you as well as by SCOPE/HUSTLER. To minimize the amount of central memory used by a job, SCOPE/HUSTLER automatically selects the the field length used to execute all system programs. User programs are loaded at a user-specified field length, but after loading, the field length is reduced to the minimum necessary to execute the program. You can, and in some instances, must, override the system controls for certain types of programs.

The job field length is defined by a starting address, called the **reference address** (RA), and the number of words in the block, which — like the block itself — is called the **field length** (FL). These values are held in the control point area and are loaded into the RA and FL registers of the CPU when it is assigned to the job at that control point. Every reference to a memory address by a central processor program is relative to the current reference address. For example, a user's reference to location 100 is actually a reference to location RA+100. The reference address enables SCOPE/HUSTLER to relocate a program without changing references within the program. The field length is also variable, depending on the program being executed.

The pool, as described in Section 1.2.2, is part of the locally developed scheduling structure. The pool is a logical extension of the control points. Just as the control points consist of all jobs that can be scheduled to the CPU, the pool consists of all jobs that can be scheduled to a control point.

In other words, there are two levels of scheduling: the unified scheduler determines which jobs are to be multiprogrammed, while the Monitor determines how the central and peripheral processors, the data channels, and the magnetic disks are to be shared among those jobs.

The criteria used to select jobs for execution at a control point are maintained within an area of central memory known as the **pool table**. Every quarter second, the scheduler scans this table and evaluates a priority formula for each job either ready to execute, or executing, at a control point. If necessary, the scheduler will "swap out" a job assigned to a control point in order to allow a higher priority job to execute. Thus, a job is likely to be assigned to several control points before terminating. When a job is swapped out, the contents of its control point area are saved in ECS and the entire job field length is written to either disk or to ECS. The pool provides a means for keeping track of this information in much the same way that the control point functions while the job is swapped in.

The pool contains both batch and interactive jobs. An interactive job enters the pool as soon as you log in and remains there for the duration of the interactive session. Batch jobs enter the pool from the input queue according to:

1. the priority calculated from the rate group;
2. job cost limit, central memory and MT/NT control parameters of the job card; and
3. from the "age" of the job within the queue.[1]

Within the pool, a job may be in any of three states:

1. waiting for a certain condition to be satisfied,
2. ready to execute (i.e., ready to "swap in"),
3. executing at a control point.

If a job is executing at a control point, it will be swapped out and placed in the "wait state" for any of the following conditions:

1. The job has requested a tape, but not enough tape units are currently available to meet its tape unit reservation.
2. The job has requested a tape, but the operator has not yet mounted and assigned it.
3. The job has attempted to attach a permanent file which is currently assigned to another job and multiread access is not possible.
4. The job has made a request which requires the operator to enter a command (e.g., GO) before the job can proceed.
5. The job is an interactive job waiting to receive a command or program input from the terminal.
6. The job is an interactive job waiting for the interactive system to finish transmitting output to the terminal.
7. The job is being held out by the operator.

[1] Reduced-rate jobs (RG1) will not be admitted to the pool until 5:00 p.m. each day.

8. The job is an interactive job waiting for a Front-end command reply.

9. The job is waiting for a common disposition.

10. The job is waiting for time to pass (status, repeat/reread).

When the wait condition is satisfied, the job automatically becomes eligible to swap in; it enters the "ready to execute" state. A job may also be swapped out and placed directly in the ready state if it requests more central memory than is currently available, if a higher priority job must swap in, or if the operator requests to have the job swapped out. The factors used to select which jobs will be swapped in have been carefully chosen and weighted in order to balance three needs: adequate response time to the interactive users, optimum use of central memory, and equal distribution of machine utilization among all jobs within the pool.

1.4.4 Exchange Jumps

An exchange jump consists of exchanging the contents of the CPU's registers with the contents of a specified area of central memory. This enables the operating system to reassign the CPU from one program to another without losing its place in any of the programs. Execution of a CPU program is initiated by an exchange jump. An exchange package is that area in which the contents of the CPU registers are saved when the CPU is released and from which the registers are restored when the CPU is reassigned. The particular program is defined by the contents of the exchange package area before the exchange jump took place. In order for the program to execute, the proper contents of its operational registers must be loaded into the CPU. These contents are what is contained in the exchange package area associated with the program.

1.4.5 Registers

The registers, temporary storage mechanisms, are grouped to provide three types of functions:

1. A registers (A0 through A7) contain the addresses of words in central memory.

2. X registers (X0 through X7) contain operands used in calculations, and the results of calculations.

3. B registers (B0 through B7) have no connection with central memory. B registers generally provide means for program indexing.

When an address is placed in any register A1 through A5, the contents of that address in central memory are read into the corresponding operand register X1 through X5. When an address is placed in register A6 or A7, the word in the corresponding operand register X6 or X7 is stored in that address in central memory.

Registers X1 through X5 hold operands read from central memory for use in calculations, and registers X6 and X7 hold results of calculations to be sent to central memory. The operands are manipulated by the arithmetic section of the central processor.

The A0 and X0 registers are used for scratch purposes or intermediate results, or for executing instructions communicating with ECS. They have no functional connection with central memory or with any other registers.

The B registers are used for counters for such functions as program indexing. The B registers have no functional connection with central memory. The B0 register is of particular interest; it provides a constant 0 which can be used for tests against zero.

1.4.6 Program Requests

Certain operations, such as the transfer of information between central memory and an external device, cannot be performed directly by the user's program, but must be requested of the operating system. Although you may not be aware of it, each read, write, or rewind statement causes the program to store a program request in location 1 of the job field length (RA+1). Few users issue program requests directly in this manner because higher level procedures are provided by the programming language or, for COMPASS programmers, by system macros. In addition, requests made to the operating system in this manner are not supported by the Computer Laboratory as changes to the operating system may make the particular procedure obsolete.

Input/output procedures are provided for reading and writing files in either binary or coded mode, positioning files forwards or backwards, and accessing file sections in either a sequential or indexed ("random-access") manner. Other system requests may be used to generate dumps, to obtain the current time and date, and to obtain various information maintained for job control (e.g., the accumulated CPU time).

1.5 SCOPE/HUSTLER Maintenance

The following features are implemented by the Computer Laboratory for system and file maintenance.

1.5.1 File Backup

Usually during the early morning hours of each scheduled production day, or at the end of scheduled weekend production, the most current version of each permanent file, that has been created or changed since the last permanent file back-up, is dumped (copied) to tape. The Computer Laboratory backup copy of a permanent file will be up to one day old. In case of accidental destruction you can reload permanent files from the Computer Laboratory tape within a two week period. After two weeks, it is possible that a copy of the file exists although it may not contain your most recent modifications. To locate such a copy, contact the consultants.

Another set of tapes is used to hold backup copies of permanent files purged by the Computer Laboratory. Usually at the end of the production day (not weekends), each permanent file is examined to determine if:

1. the file's retention period has expired,

2. the owner's problem number has expired, or

3. the owner's problem number or ID dollar balance has been expended.

Files that fall into any of these categories are dumped to tape and then purged. While these backup files are maintained primarily as insurance against a serious system failure, they also protect you against personal errors. Files that have been dumped and purged are kept for a minimum of 14 days. You can locate the visual reel name (VRN) of the purged files using PFLIST and can recreate the file by using the PFLOAD control statement (see Section 5.4). A permanent file cannot be dumped if it is attached to a user's job. Interactive users will be notified prior to any dump operation so that they can return files they wish to have dumped.

In the event of a system failure or user error which results in the loss of permanent files, normally only the files created or the changes made during the current production day should be lost. All other files can be recreated from the backup tapes.

Permanent files are discussed in more detail in Chapter 5.

1.5.2 Deadstarts

An operating system is like an enormous, continuously-running program that reads in user jobs and produces sets of program listings, messages, maps, and dumps. Like any program, the operating system must at some time be loaded into the computer's storage facilities and receive control of the processing units. This procedure is called a **deadstart**. Several deadstarts may be used to load SCOPE/HUSTLER: recovery deadstarts, normal deadstarts, last ditch recovery deadstarts, and initial deadstarts.

Recovery Deadstarts

A recovery deadstart is usually sufficient to restore the operating system after a crash; it reloads certain PP-resident programs and reconstructs certain tables from a disk-resident copy of the system library. Jobs that are in the pool and assigned to a control point are returned to the input queue (rerun). Jobs that are in the pool and swapped out are rerun only if they were using a magnetic tape, and the tape was unloaded during the deadstart procedure. Therefore, the only jobs that should be lost in a recovery deadstart are those which were executing at a control point and for which the user had specified 'RERUN,OFF.' (see Section 7.12.7).

After the recovery, interactive users are all disconnected and must log in again. If you log back in within two hours after production resumes, the local files that were in use in your earlier session will be automatically reassigned to your terminal. Your permanent files must be reattached.

Normal Deadstarts

A normal deadstart is performed prior to every production period and, when necessary, after "fatal" crashes. The normal deadstart program copies the entire system library from magnetic tape to a disk file. It then loads appropriate programs and data structures (tables) from the disk file into the peripheral processor memories, central memory, and ECS. By setting up a new File Name Table (see Section 4.6.3), the normal deadstart program destroys all previously existing jobs. During all scheduled production shutdowns, and whenever a normal deadstart appears imminent, the operator will attempt to return all batch jobs in the pool to the input queue and then save the input and output queues. These jobs will be re-entered when production resumes. Interactive users, when warned of the shutdown, should catalog (make permanent) any temporary files that they wish to retain for another session.

Last Ditch Recovery

The last ditch recovery is used when recovery deadstart has failed. It begins much like normal deadstart. After the deadstart is complete, a program is run which attempts to reconstruct, from a tape that contains a dump of CM and ECS, the input and output queues as they existed before the deadstart. This process was developed to recover the input and output queues from loss in the event of a system failure from which recovery is not possible. A last-ditch recovery can only recover jobs on the input and output queues and local files that interactive jobs had at the time of the crash. Any batch jobs that were in execution at the time of the crash (there may be up to 20 of them) will be lost during a last-ditch recovery.

Initial Deadstart

An initial deadstart resets the entire permanent file catalog to a clean slate, destroying all files on disk at the time it is done. Initial deadstarts are never done routinely, but they may be necessary after a major catastrophe, such as a destructive failure of a critical disk storage unit. In order to restore permanent files after an initial deadstart, the Computer Laboratory maintains backup copies of all permanent files on magnetic tape. Scheduled initial deadstarts and the dumping and reloading of permanent files should go unnoticed by users. If a system failure requires an unanticipated initial deadstart, permanent files will be restored to the state in which they existed at the end of the previous production day.

2

Authorization and Accounting

2.1 Introduction

Access to the 6500 system is controlled by an authorization file containing the account numbers, names, and passwords of every authorized user. To become authorized, the prospective user must apply for a Computer Laboratory problem number (PN), through which all computer use and supply charges will be billed. Once a problem number has been assigned, several subaccounts can be set up for the user and other persons as specified by the user. Thus, there are two levels of authorization: (1) problem number managers, who receive authorization from the Computer Laboratory, and (2) users who receive authorization from a problem number manager.[1]

The introductory sections of this chapter explain the central concepts of the authorization system. They describe how to apply for a problem number; they describe the identifiers, dollar balances and other limits associated with the PN; and they describe the controls available to the PN manager.

Subsequent sections describe how the Authorization File utility, AUTHORF, is used to manage the PN and to protect the user's account from unauthorized use. The final sections give a detailed description of how job costs are computed and recorded.

2.1.1 Eligibility

Authorization for computer services is available to persons engaged in University-connected research and to instructors who wish to use the computer in their classwork. A ruling of the Board of Trustees allows the Computer Laboratory to accept work from outside the University only "where there is no conflict with either on-campus teaching or research."

2.1.2 Applications

To receive a problem number, the prospective user must complete and file an "Application for Computer Services" with the Computer Laboratory Main Office. This form is available from many departmental offices as well as the Computer Laboratory office.

This form requires the following information:

a) *Applicant's name (this will be used for the "master ID")*

[1] Other levels of authorization will be discussed in Section 2.6.1.

b) Billing information (University account number, department, college, etc.)

c) Type of work (faculty, graduate, undergraduate, class, administrative, etc.)

d) A brief project description

e) Number of dollars authorized for computer use

f) Project completion date

g) Signature of the sponsoring department chairperson or a representative authorized to sign for the given University account number.

h) Specification of any special requirements

If interactive or MERIT Network services are desired, these should be so indicated. If the applicant wishes to submit jobs or print output at any of the remote batch terminals, signatures of approval must be obtained for these sites. Applicants should also specify any special requirements (such as large print or time limits) anticipated for their jobs. A list of the default maximum limits may be found in Section 2.6.4.

2.1.3 Problem Number, User Name, and Password

Once the application has been approved and processed, the new user is assigned a problem number (PN), user name (ID), and password.

PN A Computer Laboratory account number through which the user or the user's department will be billed for use of computer services and supplies.

ID An identifier for the user's subaccount. The ID initially issued with the PN is called the master ID and identifies the PN manager's subaccount. Any other subaccounts subsequently created by the PN manager (see Section 2.1.7) must be identified by a unique ID.

Password A 1-10 character secret code which protects the ID from illicit use. Usually the initial password is identical to the ID. For security, the password should be changed immediately, using the AUTHORF procedure described in Section 2.2.

New users also receive several problem number cards (PNCs) which they will need to identify their account numbers to the SCOPE/HUSTLER operating system. The PNC contains the applicant's name, the PN, and the PN expiration date punched on a special card stock. Each deck submitted for batch processing must contain a PNC, Job Card, and Password Card[1] specifying the user's problem number, ID, and password respectively. The format and position of these cards are described in Chapter 3.

The password is always required for interactive access. To extend password protection to batch jobs, the PN manager (master ID) must use AUTHORF to set the PWRQD (password-required) field to ON. This procedure is explained in Section 2.4.2.

[1] The necessity of the password card is determined by the PN manager.

2.1.4 Dollar Balances

As an aid to users who must stay within a fixed computing budget, computer services to all problem numbers and subaccounts are limited by the dollar balances. Computer use charges are deducted from the dollar balances generally on a job-to-job basis (see Section 2.7 for more details). Once a dollar balance has been reduced to zero or less, all services related to that balance are terminated. Two different dollar balances are important to the user.

PN Balance — The PN balance is a limit on all services charged to the problem number. This balance is initially set to the amount stated in the Application for Computer Services. When the PN balance has been exhausted, service to all subaccounts of the PN is terminated until additional dollars have been authorized by the chairperson of the sponsoring department.

User Balance — Each subaccount of the problem number is limited by a user (ID) balance. If a user balance is depleted only service to that subaccount is terminated. Except for the master ID, the PN manager sets user balances when the subaccounts are created; they may be reset by the PN manager at any time.

2.1.5 Resource Limits

All users of the PN are limited in the amount of central memory, central processor time, and other system resources they may request for their jobs. Actually, the user must be concerned with three levels of limits. The first are job limits, which the user specifies on the job card (see Section 3.2.3) to limit the resources available to that particular job. Next are the PN limits, which specify maximum and default values for the job card parameters. The PN limits are recorded in the 6500 Authorization File and apply to all subaccounts of the PN. Third are the maximum limits, which are the maximum values to which the PN limits can be set by the PN manager. See Section 2.6.4 for a list of these limits.

2.1.6 Job Authorization

The problem number, its IDs, passwords, and limits are recorded in a disk-resident permanent file known as the Authorization File (or AF). Before allowing a job to execute, SCOPE/HUSTLER checks the Authorization File to verify that:

1. the PN is authorized
2. the specified ID is authorized for the PN
3. the specified password is correct (if required)
4. the PN and ID dollar balances are positive
5. the requested job limits are within the authorized PN limits, and the requested job cost limit is within the current ID and PN dollar balances.

If a job fails any of these tests, it is aborted immediately. During execution, authorization data is used to determine whether certain types of control cards are authorized, and whether certain resource requests are within the PN limits.

2.1.7 Master ID vs. User ID

The user name initially issued with a new problem number is called the master ID for that PN, and the user to whom it is assigned is called the problem number manager. After receiving their problem numbers from the Computer Laboratory, PN managers can set up subaccounts for themselves and other persons by executing the AUTHORF program. Each subaccount is identified by a unique user name (ID), limited by an individual dollar balance, and protected with its own password. A user authorized by the PN manager cannot create additional subaccounts, because only the master ID is allowed to perform those functions with AUTHORF. Each problem number, therefore, has one and only one master ID but may also have up to 4094 other IDs.

After a job has executed, its dollar value is subtracted from both the user's PN balance and the ID balance. The PN manager should note that the master ID balance will always be equal to the PN balance. This allows the PN manager to monitor the actual PN balance.

In addition to creating IDs, AUTHORF enables the PN manager to delete IDs and to alter their dollar balances. It also allows the resetting of many of the PN limits providing the PN manager does not exceed the maximum values set by the Computer Laboratory. To increase the maximum PN limits, the PN manager must contact the Computer Laboratory Main Office.

All users may obtain a listing of the current PN limits by executing the AUTHORF routine described in Section 2.3. But non-PN managers cannot execute AUTHORF to change their limits or to increase their dollar balances. The only authorization data that a non-PN manager may change is the password as described in Section 2.2.

2.1.8 PN Renewal

PN managers may increase their maximum limits or authorize additional input/output sources by making a request, either in person or by phone, to the Computer Laboratory Office. To be authorized for a remote batch terminal, they must also have a signature of approval from the terminal representative. Requests to increase the PN dollar balance or to extend the PN expiration date must be written and must include the same approval signatures that accompanied the original PN application. These requests may be in the form of a memo, noting the problem number, new expiration date, and new computer use dollar limit.

2.1.9 SEED

Columns 61-80 of the problem number card contain a series of "check digits," which are used to verify that the card has not been altered. When SCOPE/HUSTLER attaches the Authorization File and checks the PN, ID, and password, it also fetches the problem number SEED value and combines it with the data read from the left side of the PNC (the PN, the PN holder's name, the problem type, and the expiration date) to calculate the check digits. If these calculated check digits do not match those read from columns 61-80 of the card, the job is aborted with the message

```
INVALID PROBLEM NUMBER CARD
```

When a PN is renewed, the Computer Laboratory normally invalidates all of its outstanding problem number cards by changing the SEED value. Once the SEED has been changed, the calculated check digits will no longer match the check digits punched in the old PNCs. Users who request a renewal should therefore anticipate at least a 12-hour delay while new PNCs are punched.

To avoid this delay, users may request that their SEEDS not be changed. But in this case, the old PNCs will never expire. As long as the Authorization File contains the old SEED value and a valid PN expiration date, the old PNCs will continue to authorize jobs even though they contain an invalid expiration date.

2.2 Changing the Password

The AUTHORF CHANGE command enables all users to change their passwords.

```
AUTHORF,CHANGE,PW=password.
```

AUTHORF expects the new password to follow the PW= keyword. The new password consists of 1-10 alphabetic or numeric characters.

If the PWRQD field of the Authorization File is set to ON and password cards are handled with care, it is extremely difficult for anyone to tamper with a user's account. A user can determine whether his or her ID has been used illegally by another person by regularly checking the last access information printed at the beginning of each interactive session and in the dayfile of each batch job. Because AUTHORF provides protection for the privacy of each user's account, users are responsible for all services charged to their IDs.

If a user forgets the password for his or her ID, the problem number manager may delete and recreate the ID. If the problem number manager forgets the password for the master ID, the Computer Laboratory Office should be contacted.

```
PNC
JONES,JC100.
PW=SECRET                          current password
AUTHORF,CHANGE,PW=VERYSECRET.      new password
6/7/8/9
```

For users of the interactive system the password can be altered using one of the four methods below.

1. Type 'AUTHORF,CHANGE,PW=password.'

2. Type 'AUTHORF,CHANGE,PW.' The system will respond by blacking out 10 spaces over which the user may enter the password.

3. Type 'AUTHORF,CHANGE,PW,VETO.' This method allows the user to confirm or deny the accuracy of the typed password. AUTHORF will prompt for the password, which the user then types over 10 blacked out spaces. AUTHORF then echoes the entered password, and follows it with a question mark. The user then types Y (Yes) or N (No), indicating whether the password is correct. AUTHORF then blacks out the echoed password and, if N was entered, prompts for a new password. If Y was entered the password is accepted.

4. When in full duplex mode (without echo-back), type

   ```
   AUTHORF,CHANGE,PW=password,VETO.
   ```

   This will cause the password that was just typed to be echoed back to the terminal; the system will then prompt for verification. The user responds by typing Y(Yes) or N(No), after which the password is blacked out. Note that under MERIT Hermes echoback defaults to ON, so one of the other methods should be used.

NOTE: In methods 3 and 4, V may be typed instead of VETO.

2.3 Displaying a User's Authorization Status

Information about the current status of a user ID account can be displayed by using the AUTHORF DISPLAY command. This command enables any user to display the current dollar balance, number of runs, and last access information (date, time, and source of input) for the user's ID and/or a list of job limits common to that problem number.

To display the dollar balance, date and time of last access, and all other information specific to the user ID (see Section 2.6.2), the following command is used:

```
AUTHORF DISPLAY
```

To display all applicable job limits, use:

```
AUTHORF DISPLAY LIMITS.
```

Or, to display both PN limits and user specific information, use:

```
AUTHORF DISPLAY ALL.
```

The AUTHORF DISPLAY command has several other options for the advanced user. See Section 2.5.4 for a detailed description of DISPLAY.

2.4 Managing a Problem Number

When a problem number is initiated by the Computer Laboratory the ID associated with it is designated as the master ID. The master ID has the ability to establish additional user IDs for the problem number and to control their respective dollar balances. The master ID also has the authority to control the job resource limits imposed upon all jobs submitted under the problem number. These tasks are accomplished by the AUTHORF utility. Specifically, AUTHORF allows the master ID to add and delete user IDs, change limits and various user ID parameters, and generate reports detailing the current status of the problem number and any IDs associated with it.

Commas and spaces in the following examples are interchangeable. That is, a space is equivalent to a comma as legal punctuation between items.

2.4.1 Adding New User IDs

Several methods may be used to add new user IDs. Some of these are examined below.

Adding one or more user IDs with the same dollar balances and limits can be accomplished using a single control card. For example, to add three user IDs named MAX, HENRY and SUE, each with a dollar balance of $50 and each with initial passwords that are the same as the ID, the following statement could be used:

```
PNC
master ID,...
AUTHORF,ADD IDS=MAX/HENRY/SUE DBAL=50.
6/7/8/9
```

If you want the passwords to be different from the user IDs, you could use the following three statements:

```
PNC
master ID,...
AUTHORF,ADD ID=MAX,DBAL=50,PW=SUPER.
AUTHORF,ADD ID=HENRY,DBAL=50,PW=NOVA.
AUTHORF,ADD ID=SUE,DBAL=50,PW=GALAXY.
6/7/8/9
```

Different dollar balances could also have been specified for each of the IDs.

The above method is obviously cumbersome when many IDs (as with a large class) are involved. The following method simplifies this procedure.

```
PNC
master ID,...
AUTHORF,ADD IDS FROM INPUT,DBAL=50,PW=RANDOM.
7/8/9
SN436621
SN471132
SN612300
SN570137
SN515233
SN525213
SN526101
SN...
SN...
6/7/8/9
```

The above will add the IDs read from the INPUT file and assign each a dollar balance of $50 and a different, randomly generated password. The password for each ID will be automatically reported on the OUTPUT file.

If different dollar balances or individually assigned passwords are desired, the following might be used:

```
PNC
master ID,...
AUTHORF,ADD IDS PW DBAL FROM INPUT.
7/8/9
S436621,MICROWAVE,55.
S471132,MASTICATE,30.
S612300,ALUMINUM,45.
S570137,BACHISBACK,10.
S...
S...
6/7/8/9
```

In some cases you may want to generate IDs having a common alphanumeric prefix and a unique numeric suffix, incremented by one for each ID. AUTHORF accomplishes this with one statement, e.g.,

```
PNC
master ID,...
AUTHORF,ADD IDS=USER1 TO USER437,DBAL=25,PW=RANDOM.
6/7/8/9
```

This will add 437 different IDs (i.e., USER1, USER2, USER3,... USER437), each with a randomly generated password and a dollar balance of $25.

2.4.2 Changing Limits and Dollar Balances

The initial job resource limits are rather low. The master ID may increase or decrease these limits by using AUTHORF,CHANGE.

For example, to change the central memory limit (CM) to 60000 octal and to change the card punch limit (C) to 8 you could execute the following job.

```
PNC
master ID,...
AUTHORF,CHANGE CM=60000,C=8.
6/7/8/9
```

To subsequently change these to 120000B and 5000 respectively, you would use:

```
PNC
master ID,...
AUTHORF,CHANGE CM=120000,C=5000.
6/7/8/9
```

You can also set any value to its maximum by using the word MAXIMUM or MAX:

```
AUTHORF,CHANGE CM=MAXIMUM.
```

AUTHORF,CHANGE may also be used to alter items associated with specific user IDs (assuming they have already been created.) For example, to change a user's dollar balance to $35, use:

```
PNC
master ID,...
AUTHORF,CHANGE ID=SUE,DBAL=35.
6/7/8/9
```

If you want to increment a dollar balance by, say $20, you would use:

```
PNC
master ID,...
AUTHORF,CHANGE ID=SUE,DBAL=DBAL+20.
6/7/8/9
```

Such changes can also be specified for several users by saying:

```
AUTHORF,CHANGE IDS=MAX/HENRY/SUE,DBAL=DBAL+15.
```

or all users (excluding the master ID) by saying:

```
AUTHORF,CHANGE IDS=ALL,DBAL=DBAL+15.
```

A group of user ids may also be specified using the construct 'user-id1 TO user-id2.' This construct specifies a range of IDs starting with the user-id1 and including all subsequently created IDs up to and including the user-id2. For example, the following command resets the dollar balances for the IDs MAX and SUE, and all IDs created between them.

```
AUTHORF,CHANGE IDS=MAX TO SUE,DBAL=75.37.
```

Detailed description of AUTHORF,CHANGE can be found in Section 2.5.6.

2.4.3 Deleting User IDs

When the master ID wishes to close out a user ID's account, it may be removed from the problem number by using AUTHORF,DELETE.

To remove a single ID use:

```
PNC
master ID,...
AUTHORF,DELETE,ID=SUE.
6/7/8/9
```

To delete a list of user IDs use:

```
PNC
master ID,...
AUTHORF,DELETE,IDS=MAX/SUE/SEYMOUR.
6/7/8/9
```

To delete all user IDs other than the master ID use:

```
PNC
master ID,...
AUTHORF,DELETE,IDS=ALL.
6/7/8/9
```

See Section 2.5.8 for more uses of AUTHORF,DELETE.

2.4.4 Generating a Report

It is often desirable to generate a report giving the status of all current user IDs or the current values of all job limits. AUTHORF,DISPLAY performs these functions.

To obtain a listing of all user IDs with their current dollar balance, number of runs, and last access information, use:

```
PNC
master ID,...
AUTHORF,DISPLAY IDS=ALL.
6/7/8/9
```

The above lists the IDs in order of their creation. To obtain a listing in alphabetic order use:

```
PNC
master ID,...
AUTHORF,DISPLAY IDS=ALL,SORTED.
6/7/8/9
```

The master ID generates a more complete listing of current job limits than an ID by using:

```
PNC
master ID,...
AUTHORF,DISPLAY LIMITS.
6/7/8/9
```

See Section 2.5.4 for a detailed description of AUTHORF,DISPLAY.

2.4.5 Creating an Auto-Exec File

The automatic execution process allows a PN manager to set up a program or control statement sequence that will automatically execute whenever a user under that problem number logs in or runs a batch job. This is useful when a program or sequence of control statements is written for a novice or non-technical user; it may also be used to restrict the use of computer resources to the intended purpose. Auto-exec files are also used by experienced users to initialize their job environment: attaching files, setting up a global library set, etc. Execution of the program may be made optional or mandatory; this decision affects all users of the problem number except the PN manager.

The information to be used at the start of a job is called the initialization file. The PN manager alone has complete control of the initialization file. AUTHORF directives may be used to catalog and control its use, as below:

```
AUTHORF,CHANGE,BINIT TO LFN=WORK, PW=TODAY, REQUIRED.
```

In the above example, the batch initialization file to be cataloged is found on the local file named 'WORK'. The permanent file has the turnkey password 'TODAY'. Since the REQUIRED option is given only the PN manager can suppress execution and any attempt by the user not to use the initialization file will cause the job to abort.

A batch initialization file can be discontinued with the 'AUTHORF,CHANGE' statement,

```
AUTHORF,CHANGE,BINIT,OFF.
```

The batch initialization file can be restored with:

```
AUTHORF,CHANGE,BINIT,ON.
```

Neither ON nor OFF affects the permanent file or any of the options or password.

Note: If BINIT or IINIT is specified in the 'AUTHORF,CHANGE' statement, no PN or user fields may be changed in the same statement.

Below is an example of initialization file use in a batch job.

A user wishes all batch jobs executed under a particular problem number to be sent to the attended queue. This user also has a user library containing a number of programs which will typically be used in batch jobs for that problem number.

```
PNC
job card
PW=password
COPYBR,INPUT,CCSEC.
AUTHORF,CHANGE,BINIT,LFN=CCSEC.
7/8/9
DISPOSE,**,H.
ATTACH,ULIB1,MYUSERLIBRARY,PW=SECRET.
LIBRARY,ULIB1.
RETURN,INITFIL.
6/7/8/9
```

For additional information on auto-exec, see Section 2.5.7 and Section 7.1.3. Chapter 9 in the *Interactive System User's Guide* gives additional examples for interactive use of auto-exec.

2.5 AUTHORF

AUTHORF is a utility for manipulating and displaying the contents of the Authorization File. The items that may be altered and/or listed are determined by the user's level of authorization (i.e., Computer Laboratory manager, College level manager, department level manager, consultant, problem number manager, or user), as described in Section 2.6.1. This section describes in detail the AUTHORF language, its abilities and limitations.

AUTHORF Directives

AUTHORF processes a command language consisting of the following directives.

ADD adds new user IDs to a problem number.

DELETE removes user IDs from a problem number.

CHANGE changes job resource limits, passwords, and user dollar balances.

DISPLAY displays selected fields of the Authorization File for a particular problem number, ID, or set of user IDs.

USE directs AUTHORF to use an account other than the one being used to run the job.

END terminates AUTHORF.

The AUTHORF directives were designed with an emphasis on flexibility.

Punctuation rules are considerably looser than for most other system utilities. Commas, spaces, and all delimiter characters other than the period, slash, and equal sign are interchangeable.

Most directive verbs, keywords, and AF field names have several synonyms and abbreviations, as listed in Section 2.6.3. Alternative keywords allow the user to produce either English-like or SCOPE-like statements, such as.

```
CHANGE DBAL TO 30.00 USING IDS FROM TAPE1.

          or

CHANGE,DBAL=30,FIELDS=ID,I=TAPE1.
```

AUTHORF can take directives from the AUTHORF control card, from the INPUT file (or terminal), or from an alternate file named on the AUTHORF control card. Similarly, directives can take input from parameters in the directive statement, from the INPUT file, or from an alternate file named in the statement. See the discussion of deck structure and input files in Sections 2.5.3 and 2.5.12.

2.5.1 Definitions

This section defines terms used to describe syntactic elements common to different AUTHORF directives.

Notation

upper case words

All items in upper case are keywords and must be typed as they appear in the definition. Contrast with lower case words.

lower case words

All items appearing in lower case denote items to be supplied by the user as defined in the explanation of the command format.

special symbols

The special symbols = . * * + and ** are to be treated as upper case characters described above.

brackets [ ]

Items enclosed in brackets are optional. If more than one item appears within a pair of brackets with each item delimited by vertical bars (|) then optionally one but only one of the bracketed terms may appear.

vertical bar |

This indicates a choice between items. Only one of the items given may be chosen. Used only to separate items enclosed in brackets or braces.

braces { }

This indicates that one and only one of the enclosed items must be selected. The items are separated by vertical bars.

ellipsis [...]

This indicates that the immediately preceding construct may be optionally repeated any number of times.

delimiters

All items must be separated by one of the following delimiters: , ( ) ; : = / and with the exception of = and / all of these delimiters are treated as equivalent to a space and may be used optionally to improve readability. The equal sign and slash are used as special delimiters and must appear only as specified in the syntactic definitions.

continuation lines

If a line ends with a comma, slash, open parenthesis, colon, semicolon, or equal sign, AUTHORF treats the next line as a continuation of the statement.

af-field

Any field name or abbreviation as given in 2.6.3.

af-field-list

One or more af-field names separated by delimiters.

id

A set of 1 to 10 characters beginning with a letter and containing only letters and digits thereafter which is to be used as a new or existing user ID name. An id may not be any of the reserved names given in Section 2.6.3.

id-list

1) One or more IDs separated by equal signs or slashes, e.g.,

SUE/HENRY/JOE

2) id-1TOid-2 which refers to the IDs 'id-1' and 'id-2' and all IDs created chronologically between them. Thus the first ID listed must have been created before the second ID listed.

When used with the ADD directive, the IDs must have a common alphabetic prefix and a numeric suffix, where the numeric suffix of the first ID is smaller than that of the second ID.For example: USER1 TO USER500.

3) Any combinations of (1) and (2) separated by equal signs or slashes, e.g.,

SUE/HENRY TO BILL/JOE

4) The keyword ALL

lfn

A SCOPE local file name. The name must not exceed 7 characters; the first character must be a letter, and the remaining characters may be either letters or numbers.

password

A set of 1 to 10 alphabetic or numeric characters representing a user's password.

problem-number

A 6 or 7 character problem number beginning with the two character department code followed by a 4 or 5 digit number.

value

Any legal value that a given af-field may take on. This includes the special values of *, **, DEFAULT and MAXIMUM.

2.5.2 AUTHORF Control Card

The AUTHORF control card has two forms. The following form would be suitable when only one AUTHORF directive is to be executed:

```
AUTHORF statement.
```

statement

Any of the legal AUTHORF statements: ADD, CHANGE, DELETE, or DISPLAY. The USE and END statements would be meaningless. The control card may be continued on as many cards as needed. A continuation card is expected if the current card ends with a comma, slash, equal sign, or other non-blank delimiter. In batch runs, the last card of the set must end with a period or closing parenthesis.

When more than one AUTHORF directive is to be executed, the following form is usually more convenient:

```
AUTHORF [FROM lfn] [UPON lfn] [ABORT].
```

FROM lfn

The name of the file from which AUTHORF is to read the directives.

UPON lfn

The name of the file upon which AUTHORF is to write error messages, DISPLAY output, echoed input lines, etc. (Input lines are not echoed if this file is connected.) In other words, any output generated by an AUTHORF directive is placed on this file, unless the directive explicitly declares a different output file.

ABORT

Requests AUTHORF to abort the job in the event of fatal AUTHORF errors. Unless this parameter is specified, AUTHORF will always terminate normally, even though it may encounter errors that cause it to quit processing directives. ABORT is seldom needed since the results of an AUTHORF run are usually irrelevant to subsequent job steps.

Default Options

input file	INPUT (batch), terminal (interactive)
output file	OUTPUT (batch), terminal (interactive)

2.5.3 Deck Structure

AUTHORF may be used in one of three ways:

1) One AUTHORF statement may appear on the control card following the AUTHORF keyword. No other input is processed in this case.

2) The INPUT file may contain any number of AUTHORF statements (the user will be prompted for input from an interactive terminal).

3) An alternate file may contain any number of AUTHORF statements.

METHOD 1:

```
PNC
id,...
AUTHORF statement.
6/7/8/9
```

The statement may be any of the legal AUTHORF statements DISPLAY, ADD, CHANGE, or DELETE. The statements IF, ELSE, USE and END may not appear in this form. The AUTHORF control card may be continued on as many continuation cards as necessary. However, the last card must be terminated with a period or closing parenthesis from a batch job. Words must not be broken across cards.

EXAMPLE:

```
PNC
id,...
AUTHORF DISPLAY ALL.
6/7/8/9
```

METHOD 2:

```
PNC
id, ...
AUTHORF.
7/8/9
statements
6/7/8/9
```

In this form the AUTHORF statements appear on cards in the INPUT file. Each statement begins on a new card and may be continued on any number of continuation cards. Words must not be broken across a continuation. Using the 'AUTHORF.' command interactively will cause the terminal to be prompted for the AUTHORF statements. Optionally an UPON lfn parameter may be specified so that output will be written on a file other than OUTPUT or so that the output will not print at an interactive terminal.

EXAMPLE:

```
PNC
master id, ...
AUTHORF.
7/8/9
ADD IDS=JOE/JOHN/JIM,DBAL=25.
CHANGE CM=120000 T=500.
DISPLAY ALL FOR IDS=ALL
USE PN=016833 ID=CHUBBY PW=FARKEL.
DISPLAY LIMITS.
END
6/7/8/9
```

METHOD 3:

```
PNC
id, ...
ATTACH,lfn1,pfn.
AUTHORF,FROM lfn1,UPON lfn2.
6/7/8/9
```

This form reads card images from the alternate input file specified by FROM lfn1. The card images contain AUTHORF statements as might appear for Method 2. The output from AUTHORF may optionally be written to a file other than OUTPUT by using the UPON lfn2 parameter to specify the output file to be used.

EXAMPLE:

```
PNC
master id, ...
ATTACH,A,AUTHORFDIRECTIVES.
AUTHORF FROM A.
6/7/8/9
```

2.5.4 DISPLAY

The DISPLAY statement prints, or copies to a local file, the contents of selected fields from the Authorization File. The fields that may be displayed are determined by the user's level of authorization.

```
DISPLAY [af-field-list|ALL|LIMITS|MAXLIMITS|IDFIELDS|DEFLIMITS]
        [UPON lfn]
        [[FOR] IDS=id-list]
        [FULL|SHORT]
        [SORTED]
```

af-field-list	A list of the fields to be displayed.
ALL	All fields to which the user has read access (i.e., authorization to display) are displayed. See Section 2.6.2.
LIMITS	Current and default job resource limits are displayed. These limits apply to all users of the problem number.
MAXLIMITS	Maximum job resource limits are displayed, only to the PN manager.
IDFIELDS	ID fields are displayed.
DEFLIMITS	Default job resource limits are displayed.
IDS=id-list	The IDs for which information is to be displayed.
UPON lfn	The name of the output file.
FULL	Selects columnar format.
SHORT	Selects compressed format (does not affect format of ID information if an ID list is specified).
SORTED	Causes the ID information to be sorted alphabetically by ID name.

Default Options

fields listed	ID-FIELDS
ids	The user ID associated with the job (or last USE directive).
output file	OUTPUT (batch), terminal (interactive)
format	FULL (batch), SHORT (interactive)
order of output	ID information is displayed in the order the IDs are listed. If IDS=ALL is specified, the information will be displayed in the order that the IDs were created.

The DISPLAY statement is also used to display the status of the initialization file.

```
DISPLAY [BINIT|IINIT|INIT].
```

BINIT displays options in effect for the batch initialization file.

IINIT displays options in effect for the interactive initialization file.

INIT displays options in effect for both batch and interactive initialization files.

The value displayed is one of the following:

```
ON (NORMAL)
ON (REQUIRED)
ON (OPTIONAL)
OFF
OFF (PURGED)
```

Display Formats

The fields that apply to all users of a problem number are called the PN fields. These, if any are requested, are displayed first. In the format selected by FULL, each PN field is printed on a separate line. The field name, its current value, a units label, and a descriptive label are aligned in columns. The format selected by SHORT is identical except that extra spaces are removed and the descriptive labels are omitted.

If the IDS= parameter is omitted, the fields associated with the user's ID are displayed in the same format as the PN fields. Otherwise, the information for each ID begins on a new line, and the values for each ID field are aligned in columns.

All output is formatted to fit in a maximum of 72 columns.

2.5.5 ADD

The ADD statement adds new user IDs to a problem number.

```
ADD [IDS=id-list|[USING af-field-list][FROM lfn]]
    [UPON lfn]
    [af-field={value|MAXIMUM|DEFAULT}]
    [PWS={password|IDS|RANDOM}]
    [VETO|LIST]
```

IDS=id-list The list of IDs to be added. If the form 'id1 TO id2' is used, where id1 and id2 have a common alphabetic prefix and numeric suffixes, IDs are generated by incrementing the numeric suffix, starting with id1 up to and including id2.

EXAMPLE:

```
IDS=A/B/C          Creates 3 IDs: A, B, and C

IDS=A1 TO A12      Creates 12 IDs: A1, A2, ..., A12
```

If an ID list is specified, the FROM and USING parameters are illegal.

USING af-field-list

List of fields to be read from the input file. These must be names of ID fields that the master ID is authorized to set, such as ID, DBAL, and PW. Field values read from the input file override those specified in the ADD statement.

The keyword USING may be omitted.

FROM lfn

The name of the input file. The format of this file and its use are discussed below and in Section 2.5.12.

UPON lfn

The name of the file on which ADD writes error messages, reports randomly generated passwords, and echoes lines from the input file. Input lines are not echoed if this file is connected.

af-field=value

Assigns a value to an ID field—typically DBAL—and applies to each ID added, unless overridden by a field value assignment in the input file. The ADD statement does not allow PN field values to be changed; use the CHANGE statement instead.

The keyword MAXIMUM assigns the maximum authorized value to the specified field. The maximum for the user dollar balance is the current PN dollar balance. Section 2.6.4 lists the maximum values for other fields.

The keyword DEFAULT assigns a default value, as determined by AUTHORF, to the specified field. For most fields the default value is considerably less than the maximum value. See Section 2.6.4.

PWS=password

Specifies a password to be assigned to each ID added.

PWS=IDS

Sets the password identical to the user ID for each ID added.

PWS=RANDOM

Assigns a unique, randomly generated password to each ID added. The passwords are reported in the output file.

LIST

Causes the ID, dollar balance, and password of each new user to be displayed.

VETO

Allows the user to cancel each addition before it is actually made (see Section 2.5.13). Equivalent to LIST in batch mode.

Default Options

ids

If omitted, ADD will read IDs from the input file.

input file

INPUT (batch), terminal (interactive)

output file

OUTPUT (batch), terminal (interactive)

af-field-list	If an input file is read and the field list is omitted, the input items must be prefixed by keywords. See Section 2.5.12.
af-field = value	Default values are listed in Section 2.6.4.
password	PWs = IDS

Using an Input File With ADD

The ADD directive expects input from a file if

1. The IDS = parameter is omitted, or
2. FROM lfn is specified, or
3. USING af-field-list is specified.

If the ID list is omitted, each line of the input file should specify an ID to be added and, optionally, a dollar balance, password, or specification of any other field associated with that ID. If the ADD statement does not list the fields to be read from the input file, each item on the input file must be prefixed by a keyword just as if it were part of the ADD statement. Otherwise, the keyword prefixes may be omitted.

Examples:

```
ADD.
ID=JOE DBAL=100 PW=ABC
ID=JOHN DBAL=25 PW=DEF
ID=JIM DBAL=200 PW=GHI
END

ADD IDS DBAL PW.
JOE 100 ABC
JOHN 25 DEF
JIM 200 GHI
END
```

An ID list must not be specified along with either an AF field list or the name of an input file.

Cautions

1. Each ID must be unique within the problem number. If the user attempts to add a duplicate ID, a diagnostic is given and the ID is ignored.

2. The total number of IDs associated with the problem number is not permitted to exceed the number specified by the MAXID field.

3. The user dollar balance assigned to the new IDs is not permitted to exceed the current PN dollar balance. If it does, the PN dollar balance is substituted.

4. Field value assignments in the input file override those specified in the ADD statement. An instructor who has students prepare cards with 'ID=name, PW=password', should restrict the fields read from the input file to ID and PW, since otherwise a student could add a 'DBAL=1000.00' parameter that would override the dollar balance assigned by the instructor.

   ```
   Unsafe: ADD FROM INPUT, DBAL=20.
   Safe:   ADD USING ID PW FROM INPUT, DBAL=20.
   ```

5. A decimal point in a dollar balance value will be interpreted as a period, which will terminate the control statement. In a batch job, this can be avoided by enclosing the dollar balance in '$', e.g. 'DBAL=$75.50$.' However, this only applies to control statements not to input or alternate input files. The delimiters may be used singly (e.g. $75.50) or in pairs. The use of '$' delimiters is accepted in interactive mode. Note: The decimal point must be followed by numerical characters only.

2.5.6 CHANGE

The CHANGE statement is used to alter the contents of one or more AF fields. The fields that may be altered depend on the user's level of authorization.

Because password changes are processed specially, the syntax for this type of change is shown separately. Note that no other changes may be specified with a password change.

```
CHANGE {PW=password|[USING] PW [FROM lfn]}
       [UPON lfn]
       [VETO|LIST]
```

PW=password specifies the new password, consisting of 1-10 alphabetic or numeric characters. If this parameter is specified, the FROM and USING options are illegal.

USING PW indicates that the password is to be input from a file, or interactively, from the terminal. The keyword USING may be omitted.

FROM lfn the name of the input file containing the new password.

UPON lfn the name of the output file, upon which error messages or, if selected, the echoed password will be written.

LIST echoes the new password.

VETO echoes the new password and then permits the user to accept or reject it. VETO is equivalent to LIST for batch jobs.

Other changes to the Authorization File may be specified with the following syntax.

```
CHANGE [af-field=expression] [...]
       [[FOR]IDS=id-list [NEWIDS=id-list]
       |[USING af-field-list] [FROM lfn]]
       [UPON lfn]
       [VETO|LIST]
```

af-field=expression — Sets the specified AF field to a new value. This parameter is fully described below, under Field Change Parameters.

Note: The user should not attempt to change both PN and ID fields within a single CHANGE statement.

IDS=id-list — The list of IDs to be affected by changes to any ID fields. If an ID-list is specified, the FROM and USING parameters are illegal.

NEWIDS=id-list — A list of IDs to be substituted for those given by the IDS= parameter. This list must match in number the list following IDS=', since the i-th ID in the NEWIDS list renames the i-th ID in the IDS list.

USING af-field-list — Lists items to be read from the input file. The keyword USING may be omitted.

FROM lfn — The name of the input file if field change expressions and/or IDs are to be read from a file. Field change parameters read from the input file override corresponding parameters in the CHANGE statement. See below and Section 2.5.12 for further discussion of the input file.

UPON lfn — The name of the output file on which CHANGE writes error messages and echoed input lines. Input lines are not echoed if this file is connected.

LIST — Causes the changes to be displayed as they are made.

VETO — Permits the user to cancel each change before it is actually made. Equivalent to LIST in batch mode.

Default Options

af-field=expression — If both field change parameters and the NEWIDS= parameter are omitted, the input file will be read.

ids — If ID field changes are specified, the job ID (or the ID given in the last USE statement) is assumed. If NEWIDS= is specified, IDS= must also be specified.

input file — INPUT (batch), terminal (interactive)

output file — OUTPUT (batch), terminal (interactive)

af-field-list — If an input file is read and the field list is omitted, the input items must be prefixed by keywords.

Field Change Parameters

Field change parameters assign a new value to a particular field in the user's Authorization File entry. In its simplest form the parameter is

af-field=value

where 'af-field' is the AF field name and 'value' is a value appropriate to the type of field. The following keywords are also accepted for 'value':

MAXIMUM or MAX

> This keyword is defined only for job resource limits, where the maximum value that may be assigned to the field is determined by the MLC (maximum limits class) field. Section 2.6.4 lists the maximum values associated with each level of the MLC field.

DEFAULT or DEF or **

> This keyword, like the keyword MAXIMUM, is defined only for job resource limits. The default values are those assigned by the Computer Laboratory when the problem number is initiated. For most fields, the default value is considerably less than the maximum value. Section 2.6.4 lists both default and maximum values as determined by the MLC field.

Field values may also be incremented or decremented with the forms,

```
af-field=*+value
af-field=*-value
```

or,

```
af-field=af-field+value
af-field=af-field-value
```

If the latter forms are used, the field named on the right side of the equal sign must match that on the left side. To avoid ambiguity with the minus sign, the field name on the right side of the equal sign must not be hyphenated.

Increment and decrement operators are restricted to numeric fields and the SOURCE field.

One further embellishment of the field change parameter permits the use of 'TO' in place of the equal sign. For example,

```
CHANGE CM TO 100000.
```

is an acceptable alternative to CHANGE,CM=100000.

Octal vs. Decimal

The values of certain fields, such as CM, are traditionally displayed and specified in base 8 rather than base 10. In AUTHORF such fields are indicated by displaying the value with a B suffix. Values specified for these fields must be octal unless a D suffix is used to indicate decimal. Conversely, an octal value can be specified for a decimal field by adding a B suffix. Legal suffixes for numeric values are:

B	indicates an octal number
D	indicates a decimal number
K	equivalent to 000B

Example: 100K = 100000B = 32768D

Truncation and Rounding of Field Values

The values of some fields are scaled to fit more compactly in the Authorization File. As a result, the values specified by the user may be truncated to the next lowest, or rounded up to the next highest, multiple of the scale factor. For example, the page limit is truncated to the next lowest multiple of 8. The fields affected by such conversions are described below.

C,DC raised to multiple of 8 cards.

CM,DCM
EC,DEC truncated to multiple of 100B words.

L,DL raised to multiple of 8 pages.

CT converted from minutes to seconds and then truncated to a multiple of 8 seconds.

PNPFL,UPFL
MS,DMS raised to a multiple of 64 PRUs.

Using an Input File with CHANGE

CHANGE will read from the input file if:

1. Neither a field change parameter nor a NEWIDS= parameter is specified, or
2. FROM lfn is specified, or
3. USING af-field-list is specified.

It is illegal to combine the ID parameter with either the FROM or USING parameters.

The input file is used to supply the list of IDs to which the field change parameters in the CHANGE statement apply, or to supply a list of IDs along with field change parameters for each of the IDs. Changes to PN fields can also be specified, providing the input line does not contain an ID field.

Some examples will illustrate these applications.

```
CHANGE.
ID=JOE DBAL=*+10.
ID=JOHN DBAL=25.
ID=JIM DBAL=0.
END
```

Makes different changes to the dollar balances of JOE, JIM, and JOHN.

```
CHANGE ID DBAL.
JOE *+10
JOHN 25
JIM 0
END
```

Makes the same changes as the preceding examples, but here the addition of a field list to the CHANGE statement eliminates the need for keyword prefixes with the input items.

```
CHANGE IDS, DBAL=DBAL + 20.
JOE
JOHN
JIM
END
```

Increments the dollar balances of JOE, JOHN, and JIM by $20.

```
CHANGE DBAL=20.
ID=JOE
ID=JOHN
ID=JIM DBAL=50
CM=120000
END
```

Changes dollar balances for JOE and JOHN to $20. Changes the central memory limit, which applies to all users of the PN, to 120000B words.

Cautions

1. Only those fields for which the user has write access can be changed. See Section 2.6.2.

2. Values assigned to job resource limits cannot exceed the maximums imposed by the MLC (maximum limits class) field. Section 2.6.4 lists the maximum associated with each level of the MLC field. An attempt to assign a value higher than the user's authorized maximum will result in an informative message and the substitution of the maximum value.

3. Each ID must be unique within the given problem number. If NEWIDS= specifies a duplicate name, the corresponding ID in the IDS= list will remain unchanged. Renaming takes immediate effect, so the following name changes would be illegal:

   ```
   CHANGE IDS=A/B/C, NEWIDS=B/C/A.
   ```

4. The CHANGE statement should not specify changes to both PN and ID fields. For example, the statement

   ```
   CHANGE,CM=120000,DBAL=20,FOR ID=JOE.
   ```

   is improper because it suggests that the new CM limit applies only to JOE.

5. A decimal point in a dollar balance value will be interpreted as a period, which will terminate the control statement. In a batch job, this can be avoided by enclosing the dollar balance in '$', e.g. 'DBAL=$75.50$.' However, this only applies to control statements, not to input or alternate input files. The delimiters may be used singly (e.g. $75.50) or in pairs. The use of '$' delimiters is accepted in interactive mode. Note: The decimal point must be followed by numerical characters only.

2.5.7 CHANGE—The Auto-Exec Feature

The PN manager can use a CHANGE directive to create and change an initialization file. Actual execution of initialization files is described in Section 7.1.3.

An initialization file, information to be used at the start of a job, can be cataloged and controlled in the following manner:

```
CHANGE  {BINIT|IINIT}  [TO]  [LFN=lfn]  [PW[=password]
        [I=inlfn] [control option] [VETO|LIST].
```

LFN=lfn

is the name of a local file that contains the control statements or program binary (i.e. relocatable, absolute overlay or segment formats) to execute at the beginning of the interactive or batch job.

"INITFILEFORPNnnnnnINTERACTIVE"

or

"INITFILEFORPNnnnnnBATCH"

are the permanent file names in which the file is copied and cataloged, and where nnnnn is the problem number. Any previous file by this name is purged.

password

is the turnkey password of 1-9 alphanumeric characters used when cataloging the permanent file. Password specification allows the file to be attached (with all permissions) at times other than the start of the interactive or batch job. If no password is specified, a random password is generated, which will prevent the PN manager from attaching the file except at the beginning of the job. This situation can be altered by changing the password; for example:

```
AUTHORF,CHANGE,BINIT,PW=newpw.
```

This will cause the current initialization file to be recataloged with the password 'newpw'.

In interactive mode, if PW is given alone, the user is prompted for input. In batch, a separate input file may be specified by I=inlfn; AUTHORF will accept only a password from this file.

control option

specifies whether execution of the initialization file is optional or required. The option in effect is not changed when the file or password is modified.

NORMAL selects the use of the initialization file unless suppressed by the user. This is the default.

REQUIRED causes the initialization file use to be required at the start of every job. Only the PN manager may suppress execution; an attempt by an interactive user to suppress the use of the initialization file will prevent logging in. In batch mode, this action will prevent the job from running.

OPTIONAL will not normally execute the initialization file. The user may request it via the job card or log-in option INIT (see Section 7.1.8).

Other parameters are implemented when changing the status of the initialization file.

```
AUTHORF,CHANGE{IINIT|BINIT}[TO]{OFF|ON|PURGE}[VETO|LIST].
```

OFF discontinues use of the initialization file. The password and options do not change, nor is the permanent file purged.

ON restores use of the initialization file without changing options or the file.

PURGE terminates use of the initialization file. Also it causes the permanent file to be purged, and the password and options in the Authorization File are cleared.

If no initialization file exists and LFN is not supplied, OFF and PURGE have no effect; specifying ON will cause a fatal error.

Options VETO and LIST may be used to verify initialization file changes; for descriptions of these options, see Section 2.5.13.

2.5.8 DELETE

The DELETE statement removes existing IDs from the problem number. The master ID cannot be deleted.

```
DELETE [IDS=id-list|FROM lfn]
       [UPON lfn]
       [VETO|LIST]
```

IDS=id-list — The list of IDS to be deleted. The keyword ALL will delete all IDs except the master ID. Illegal if FROM parameter is specified.

FROM lfn — The name of the file from which a list of IDs is to be read. Illegal if ID parameter is specified.

UPON lfn — The name of the output file on which DELETE is to write error messages and echo input lines. Input lines will not be echoed if this file is connected.

LIST — Display each ID deleted.

VETO — Allows the user to cancel each deletion before it is actually made. Equivalent to LIST in batch mode.

Default Options

ids — If an ID list is omitted, IDs will be read from the input file.

input file — INPUT (batch), terminal (interactive).

output file — OUTPUT (batch), terminal (interactive).

NOTE: When user accounts are deleted and recreated, they will have new PN ordinals. All permanent files previously cataloged by these IDs will no longer be associated with them. The permanent files will remain on the system; storage charges will decrement the PN dollar balance, but will not be charged to the user.

Using an Input File with Delete

DELETE will read the IDs to be deleted from the input file if:

1. The IDS= parameter is omitted, or
2. FROM lfn is specified.

Each line of the input file should contain one ID; the prefix 'ID=' is not necessary.

Example:

```
DELETE.
JOE
JOHN
JIM
END
```

Caution

Deleting user IDs will cause any permanent files previously cataloged with these IDs to be charged to the PN dollar balance only. If the user wishes to purge the permanent files belonging to these IDs, the dollar balance should be set to a negative value for the IDs to be deleted. The deletion may then occur the following weekday or later.

2.5.9 USE

When a user wishes to examine or alter authorization information for a problem number or ID different than the one being used to run the job, the USE statement specifies that problem number, or ID, and proves that the user is authorized to access it.

```
USE PN=problem-number ID=id PW=password
```

PN=problem-number	The problem number to be used.
ID=id	A user ID or master ID.
PW=password	The proper password for the PN and ID specified.

All three parameters must be specified.

After the USE statement has been successfully executed, AUTHORF functions as if the user were logged in under the PN, ID, and password specified. For example, if the ID specified by USE is a master ID, AUTHORF grants the user the access privileges of the PN manager.

AUTHORF will not permit a USE directive to an ID which has either IINIT or BINIT set to REQUIRED by the PN manager. Exceptions are:

1. If the ID requested is the master ID for the problem number.

2. If the ID which does the USE is the master ID for the same problem number.

3. If the ID on the previous USE directive is the master ID for the same problem number.

By using this restriction, it will not be possible for any user other than the PN manager to change the password, except by logging in under that ID directly. If the initialization file is REQUIRED, and if control does not return to the user, it will not be possible for the user to change the password.

2.5.10 END

This statement causes AUTHORF to stop processing. It is equivalent to an end-of-record or end-of-file.

```
END
```

END cards are also used to terminate a set of data cards for a particular directive when directives and data cards are interspersed; see Section 2.5.12.

2.5.11 IF* and ELSE*

The IF statement allows conditional performance of other AUTHORF statements. For example, an IF statement used in conjunction with a DISPLAY statement could be used to display the user ID fields for all IDs whose dollar balances are less than $10.

```
IF {af-field {EQ|NE|GT|LT|GE|LE} value|SAME} statement.
```

af-field	any af-field the user has access to. All PNs and IDs under the control of the user will be scanned.
value	A legal value for the af-field specified.
SAME	The condition specified in the previous IF.
statement	any legal DISPLAY, ADD, CHANGE, DELETE, or END statement.

Example:

```
IF DBAL LT 10 CHANGE DBAL TO 25.
```

The ELSE statement reverses the condition of the preceding IF. That is, it may be considered as an IF that specifies the opposite condition of the preceding IF. Thus, ELSE may be used only after an IF. The 'statement' portion may be any legal DISPLAY, ADD, CHANGE, DELETE or END statement.

```
ELSE statement
```

When the 'statement' portion of an IF or ELSE contains an IDS=id-list field, then the search is restricted to the IDs indicated. The default for the IDs list (for PN managers) is IDS=ALL. The IDS=id-list is inappropriate for normal user IDs to use.

* IF and ELSE are not currently implemented.

2.5.12 Input Files

The ADD, CHANGE, and DELETE statements accept input from files as well as from parameters included in the statements. This ability is particularly useful when adding, deleting, or altering many IDs and the form 'id1 TO id2' is not applicable.

The items to be read from such files must be identified either by specifying a field name list in the statement (e.g., USING af-field-list) or by prefixing each item in the input file with a keyword, just as if the item were part of the directive. If a field list is supplied, keyword prefixes—although acceptable—are not necessary for the input items.

To illustrate, suppose you wish to add several IDs, specifying the ID, dollar balance, and password from the input file. Card images could be prepared in either of two styles:

(a) ID=JOE DBAL=25 PW=ABC
(b) JOE 25 ABC

Consider now the following ADD directives:

ADD. Cards like (a) are acceptable. Cards like (b) are not.

ADD IDS DBAL PW. Cards like (a) or (b) are acceptable.

A field list parameter is also useful for restricting the fields to be processed.

ADD IDS PW. For cards like (a), only the ID and PW items will be processed. For cards like (b), the first two fields will be processed ('25' would be read as the password).

If the name of the input file is not explicitly stated in the command, the data cards must immediately follow the command statement. When AUTHORF directives and data cards are interspersed in this manner, an END card must terminate each sequence of data cards.

Example:

```
PNC
master id, ...
AUTHORF.
7/8/9
ADD ID DBAL PW.
JOE 25 ABC
JOHN 50 DEF
JIM 100 GHI
END
DISPLAY IDS=ALL
6/7/8/9
```

If directives and data cards are not interspersed, each sequence of data lines must terminate with an end-of-section or end-of-partition.

```
PNC
master id,...
ATTACH,FILE,AUTHORFINPUT.
AUTHORF.
7/8/9
ADD ID DBAL PW FROM FILE.
DELETE IDS FROM FILE.
DISPLAY IDS=ALL.
6/7/8/9
```

where FILE contains:

```
JOE 25 ABC
JOHN 50 DEF
JIM 100 GHI
(END-OF-SECTION)
MAX
HENRY
SUE
(END-OF-SECTION)
(END-OF-PARTITION)
```

2.5.13 VETO

The VETO option allows the user to verify and then to accept or to reject the results of a ADD, CHANGE or DELETE command. In batch mode, the VETO option is equivalent to the LIST option.

After displaying the affected fields and their new values, AUTHORF prompts the user for a reply. The user chooses among the following replies (acceptable abbreviations are indicated by underlining).

YES	accepts the change
NO	rejects the change
STOP	rejects the change and stops processing of the command
CONTINUE	accepts the change and turns off VETO for subsequent changes
LIST	accepts the change, turns off VETO, and turns on LIST for subsequent changes

2.5.14 Abbreviations and Synonyms

To facilitate use AUTHORF allows many abbreviations and synonyms.

The following are accepted as being synonymous with the given AUTHORF statement types:

name	synonyms
DISPLAY	PRINT LIST WRITE

ADD	INSERT
	CREATE

CHANGE	ALTER
	REPLACE
	SET
	UPDATE

DELETE	REMOVE DROP
USE	none
IF	none
ELSE	none
END	STOP FINISH end-of-section end-of-partition end-of-information *EOR *EOF

Following is an alphabetized list of statement keywords and accepted abbreviations. Wherever a plural keyword is logical, the singular form is accepted. AF field names and their synonyms may be found in Section 2.6.3.

keyword	abbreviation
ADD	AD
ALL	none
ALTER	ALT
BINIT	BI, BATCH-INIT
CHANGE	CH
CREATE	CR
DEFAULT	DEF,**
DELETE	DEL
DISPLAY	DI, DIS, DISP
DROP	DR
ELSE	none
END	none
FIELDS	FIELD
FINISH	none
FOR	none
FROM	I=
FULL	F
GT	>

IINIT	II, INTERACTIVE-INIT
INIT	INIT-FILE, INIT-FIELDS
INSERT	IN
IDS	ID, USER, USERS, USERID, USERIDS, USERSNAME, USERNAMES
LFN	FILE
LIST	LI
LT	<
LIMITS	LIMIT
MAXIMUM	MAX
NE	none
NEW-IDS	NEW-ID, NEW-USERS, NEW-USER
NORMAL	NORM
OPTIONAL	OPT
PNS	PN, PNC, PNCS, PROBNUM
PW	PASSWORD, TK
PWS	PW, PASSWORD, USERPW, USERPWS, USERPASS
PRINT	PR
REMOVE	REM
REPLACE	REP
REQUIRED	REQ, MANDATORY, REQD
SAME	none
SET	none
SHORT	S
SORTED	SORT
STOP	none
UPON	O=
USING	USE, FIELDS=, FIELD=, F=
VETO	V
WRITE	WR
=	TO (only in af-field=expression)

Hyphens appearing in any keyword may be deleted. However, a space must not be substituted if this is done. Instead, the sections should be concatenated.

2.6 Authorization Levels and the Authorization File

This section describes the different levels of authorization and their abilities and limitations in regards to altering and displaying the various fields of the Authorization File.

2.6.1 Levels of Authorization

Currently three levels of authorization are operable. These are the Computer Laboratory Manager, the Problem Number Manager and the individual user.

The Computer Laboratory Manager sets the maximum values of services which a problem number can use. This person has the authority to change any field within the Authorization File. Certain changes desired by a lower level manager can be implemented only by the Computer Laboratory Manager.

The Problem Number (PN) Manager is the person to whom a newly created problem number is assigned by the Computer Laboratory Manager. The PN manager sets up individual user accounts under the PN, and establishes the maximum limits for services to be used by individual users. The PN manager has the authority to alter certain fields pertaining to that problem number. See Section 2.6.2 for a description of these fields.

The user has the lowest level of authorization. Users have authority to display only the fields which pertain to their user IDs, and can change only one field, the user password.

In addition to the current authorization levels, three more levels are planned for implementation; these are the College Level Manager, the Department Level Manager and the Consultant.

The Department Level Manager is a person from a certain department who is designated by the Computer Laboratory Manager to have the authority to alter certain fields in accounts associated with that department.

The College Level Manager is a person from within a college who is designated by the Computer Laboratory Manager to have the authority to change certain fields in PN accounts associated with that college.

The Consultant is a person designated by the Computer Laboratory Manager to have the authority to view any field in the Authorization File except users' passwords. This person does not have the authority to change any field.

2.6.2 PN and User Fields

The following is a list of Authorization File fields defined for each problem number, plus fields defined for each ID of the problem number. The list is divided into four groups; the first three define the various PN fields and their access authority, and the fourth describes the fields assigned to an individual user ID.

Group I — The following fields can be displayed by the individual user and the PN manager, but can be changed only by the PN manager.

Mnemonic	Meaning
ACCL	Access level; a number from 1 to 5 which determines what control statements can be executed.
C	Card limit; the maximum number of cards of punched output that can be specified by the job card C parameter.
CM	Central memory limit; the maximum number of central memory words (octal) that can be specified by the job card CM parameter or (interactively) by an RFL command.
CT	Connect time; the maximum number of minutes the user may be logged in during a single interactive session. After logging out, the user may immediately log in for another session.
DC[2]	Default card limit; the maximum number of cards that a job can punch if the C parameter is omitted on the job card.
DCM[2]	Default central memory limit; the initial field length used if the job card CM parameter is omitted. (If the job card CM parameter is omitted, the maximum job field length is either the Authorization File CM limit or 100000B, whichever is less.)
DMS[2]	Default mass storage limit; the amount of disk storage, in PRUs, that a job can use if the MS parameter is omitted on the job card.
DEC[1,2]	Default extended core storage limit.
DL[2]	Default line printer page limit; the maximum number of pages that a job can print if the L parameter is omitted from the job card.
EC[1]	Extended core storage limit; the maximum number (octal) of ECS words allowed the user.
FILEL	The maximum number of files that a job can have assigned to it at any one time.
L	Line printer page limit; the maximum number of pages of print that can be specified by the job card L parameter.
MS	Mass storage limit; the maximum amount of disk space, in PRUs, that can be specified by the job card MS parameter.
PF[1]	Permanent file limit; the maximum amount of disk storage, in PRUs, that can be occupied by permanent files cataloged under this ID.

[1] Field currently not in use and not displayable or alterable at present.

[2] When either the current job limit or default limit is requested, both items will be displayed. For the PN manager, the upper bounds for the job limit will also be displayed.

PHONE The telephone number of the person to whom the PN is issued.

PNON PN ON flag. This can be set to OFF to invalidate the PN.

PNTRP PN trap flag. This can be set ON to temporarily invalidate the PN if passwords, IDs, and PNCs are stolen.

PWRQD password-required flag. If ON, user passwords must be supplied by batch as well as interactive jobs. This flag is initially OFF. The PN manager may set it ON, but only the Computer Laboratory Manager can restore it to OFF.

RG A number which indicates the highest rate group that can be specified by the job card RG parameter.

RUNL Daily run limit; the maximum number of batch jobs that can be run from the central site each day (8:00 a.m. to 8:00 a.m.) under each ID of the problem number. Once this limit is reached, that ID cannot be used to submit any jobs (either from central site or Export/Import) or log in on an interactive terminal until the following day. If RUNL is 0, the number of runs is unlimited.

SOURCE Current authorization for Input/Output sources and special print queues. These codes are listed in Appendix E.

T The maximum number of central processing seconds that can be specified by the job card T parameter.

Group II The next group of fields may be displayed only by the PN manager. They cannot be changed by either the user or the PN manager.

Mnemonic **Meaning**

ACCN The University account number to which charges for computer services and supplies are billed.

COLM[1] Indicates the college level manager.

DEPTM[1] Indicates the department level manager.

MAXID The maximum number of IDs a PN manager can create.

MAXSOURCE[1] The maximum I/O source authorization allowed the PN manager.

MCM The maximum central memory field length that the PN manager is allowed to authorize in the CM field.

MEC[1] The maximum extended core storage limit that the PN manager is allowed to set on ECS.

MLC Specifies the set of maximum limits authorized for the PN. See Section 2.6.4.

MMS The maximum mass storage limits which the PN manager can set on MS.

[1] Field currently not in use and not displayable or alterable at present.

MPAGE	The maximum number of pages to which the PN manager can set PAGES.
MTIME	The maximum number of seconds the PN manager can authorize in the TIME field.
NID	The number of current IDs in use under a certain PN.
PNDBAL	The problem number dollar balance. This field is initially set by the Computer Laboratory, but is thereafter decremented by the system as each job is run.

Group III The third group of PN fields contains fields which are displayable by both the individual user and the PN manager, but not changeable by either.

Mnemonic	**Meaning**
CLAF	Computer Laboratory Manager flag.
DEPT	The PN manager's department code.
EXPDAT	The expiration date of the PN.
HAL[1]	HAL programmer flag.
OWNER	The name of the person to whom the PN is issued (not to be confused with the master ID).
PFTOT[1]	Total PF space in use by the PN.
PN	The problem number—an account number used by SCOPE/HUSTLER to record the use of computer services and supplies.
PT	Problem type; a classification of the type of project, e.g., graduate research, CPS class, etc., which is used for billing and Computer Laboratory's records.
SYS	System programmer flag.

Group IV The last group are fields pertaining to individual user IDs under a PN. All fields are displayable by both users and PN managers, except where noted.

Mnemonic	**Meaning**
DATELA	date of last access.
DBAL	Current user dollar balance. This value, initially set by the PN manager, is decremented by the system after each job that is run under this ID. The PN manager can increment or reset it.
ID	User ID; can be changed by the PN manager.

[1] Field currently not in use and not displayable or alterable at present.

PW — user password; this is the only field which the user can change. No one can display it.

RTODAY — Daily runs; the number of runs today (8:00 a.m. to 8:00 a.m.) for this ID.

RUNS — Total runs; the total number of runs since authorization for this ID.

SOLA — Code character representing the source of last access.

TIMELA — Time of last access.

UORD — User ordinal; a system-generated number that uniquely identifies each user ID in the Authorization File. Note: If a user ID is renamed, the user ordinal remains the same. But if the user ID is deleted and re-added, a new user ordinal is assigned.

UPFL[1] — The maximum amount of permanent file storage which can be used by this ID; can be changed by PN manager.

UPFT[1] — The amount of permanent file storage in use under this user ID.

Displaying Groups of Fields

A user or PN manager can display certain groups of related fields by using the pseudo-field names (names which refer to a group of fields) described below. Only fields to which the user has display access are included.

Name	**Fields Displayed**
ALL	All fields.
DEF-LIMITS	DCM, DL, DC, DMS
ID-FIELDS	ID, DBAL, SOLA, DATELA, TIMELA, RUNS, UORD
LAST-RUN	SOLA, DATELA, TIMELA
LIMITS	CM, T, L, C, RG, ACCL, MS, FILEL, CT, SOURCE, MAXID, PWRQD, DCM, DL, DC, DMS, MCM, MT, ML, MC, MMS, MFILEL, MCT
MAX-LIMITS	PN, PNDBAL, MAXID, MCM, MT, ML, MC, MMS, MRUNL, MFILEL, MCT, MACCL, MRG (PN manager only)

2.6.3 Abbreviations

To facilitate use of AUTHORF, many synonyms of the Authorization File field names are allowed. Below is a list of the field mnemonics, arranged in alphabetical order, followed by the acceptable synonyms.[2]

[1] Field currently not in use and not displayable or alterable at present.

[2] AUTHORF examines only the first 8 characters of the field named.

Mnemonic	Synonyms
ACCL	ACCESS-LEVEL, AL, ACC-LEVEL, ACCESS
ACCN	ACCOUNT, ACCOUNT-NUMBER, ACCT
C	CARD-LIMIT, PUNCH-LIMIT, CARDS, CARDL, PUNCH, CARD, PUNCHL
CLAF	CL-FLAG, CLM, CL-MGR, CL-MANAGER
CM	CENTRAL-MEMORY, CM-LIMIT, CMFL, CML, SCM
CT	CONNECT-TIME, CONNECT, CTL, ICONL
DATELA	DATE-OF-LAST-ACCESS, LAST-DATE, DOLA
DBAL	UDBAL, $, ID$, $ID, USER$, $USER, ID-DBAL, USER-DBAL, ID-DOLLARS, USER-DOLLARS [1]
DC	DCARD, DCARDL, DCARDS, DPUNCH, DEF-CARD, DEF-CARDS, DPUNCHL, DEF-CARDL, DEF-PUNCH
DCM	DCMFL, DSCM, DEF-CM, DEF-CMFL, DEF-CM-LIMIT, DCML, DFL, DEF-CML
DEC	DECS, DECSFL, DLCM, DEF-ECS, DEF-ECSFL, DEF-ECS-LIMIT
DEPT	DEPARTMENT
DL	DPAGE, DPG, DPAGEL, DPAGES, DPRINT, DEF-PAGE, DEF-PAGEL, DEF-PAGES, DEF-PRINT
DMS	DMSL, DEF-MASS-STORAGE, DEF-DISK, DEF-DISK-LIMIT, D-DISK
EC	ECS, ECSFL, ECSL, ECS-LIMIT, LCM
EXPDAT	EXP-DATE, EXPIRE, EXPIRATION
FILEL	FILE-LIMIT, FILE, FILES
ID	IDS, USER, USERS, USER-ID, USER-IDS, USER-NAME
L	PAGE-LIMIT, PRINT-LIMIT, PAGEL, PG, PAGE, PAGES PRINTL, PRINT
MAXID	ID-LIMIT, MAX-USER, USER-LIMIT, IDL, USERL, MAXIDS
MLC	MAX-CLASS, TBF, LIMIT-CLASS, LIMITS-CLASS, LMT-CLASS
MS	MASS-STORAGE, DISK-LIMIT, DISKL, MSL, DISK
NAME	PN-OWNER, PN-HOLDER, PN-MANAGER, OWNER, PNMGR, MANAGER

[1] Since the dollar sign ($) is a delimiter, it must appear as $$$$ (e.g. $$$$ID) in AUTHORF control cards; in AUTHORF directives, either $$$$ or '$' may be used.

NID	NUM-ID, NUM-IDS, NIDS, NUM-USER, NUM-USERS, NO-IDS, NUMUSR
PHONE	TELEPHONE, PHONE-NUMBER, PHONE-NO, TEL
PN	PROBLEM-NUMBER, PROB-NUM, PNC, PNS, PNCS
PNDBAL	PN$, $PN, PN-DOLLARS
PNON	PN-VALID, PN-ACTIVE, PN-FLAG
PNPFL	PNPF-LIMIT, PFL
PNPFTOT	PNPFT, PNPFTOT, PN-PFS, PN-PF, PN-TOTAL-PFS, PN-PF-TOTAL,PFTOT
PNTRP	PN-TRAP, TRAP-PN, TRAP
PT	TYPE, PROB-TYPE
PW	PWS, PASSWORD, USER-PW, USER-PWS, USER-PASSWORD
PWRQD	PW-FLAG, BATCH-PW, PW-REQ, PW-REQUIRED, PW-ON
RG	RATE-GROUP, PRI, P, PRIORITY
RTODAY	RUNS-TODAY, TODAYS-RUNS, RUND
RUNL	RLIMIT, RUN-LIMIT
RUNS	TOTRUN, RUN, TOTAL-RUNS, RUNT, RUNTOT
SEED	none
SOLA	SOURCE-OF-LAST-ACCESS, LAST-SOURCE
SOURCE	SOURCES, QUEUE, QUEUES
SYS	SYS-FLAG, SYSTEMS
T	CPU-TIME, TIME-LIMIT, TIMEL, TIME, TL
TIMELA	TIME-OF-LAST-ACCESS, LAST-TIME, TOLA
UORD	PORD, PNORD, PWORD, USORD, USER-ORD, USER-NUM, USER-ORDINAL, USER-NUMBER, ORDINAL
UPFL	IDPFL, USERPFL, USER-PF-LIMIT, ID-PF-LIMIT
UPFT	UPFTOT, IDPFT, IDPFTOT, USER-PF-TOTAL, ID-PF-TOTAL, UPFS, ID-PFS, ID-PF-SPACE, ID-PF, UPF, USER-PFS

2.6.4. Maximum and Default Values

Certain fields within the Authorization File represent the maximum amount of resources that a problem number can use. These fields, known as PN limits, can be established by the PN manager. The PN limits, however, cannot exceed limits set by the Computer Laboratory, known as maximum limits. The set of maximum limits controlling a particular problem number is determined by the MLC (maximum limits class) field, which is a field within Authorization File entry for that problem number.

The following table shows the set of maximum limits specified by the MLC values 1 through 7. If MLC is set to 0, the PN manager cannot change any of his PN limits.

Numeric values are decimal unless a "B" suffix is used to denote octal.

FIELD	UNIT	LIMITS						
MLC ENTRY		1	2	3	4	5	6	7
ACCL		5	5	5	5	5	5	7
CL	cards	5000	10000	20000	20000	20000	20000	32760
CM	words	60000B	100000B	120000B	120000B	120000B	120000B	170000B
CT	minutes	60	120	240	240	240	240	546
EC	words	0	0	0	0	0	0	0
FILEL	files	15	20	30	63	63	63	63
L	pages	500	5000	20000	20000	20000	20000	32760
MSL	PRUs	262080	262080	262080	262080	262080	262080	262080
PNPFL	PRUs	10000	20000	20000	20000	20000	20000	262000
RG		3	3	3	3	3	3	3
RUNL	runs	0	0	0	0	0	0	0
T	seconds	512	4096	8192	8192	8192	8192	32760

The following is a list of the initial limits assigned to every new problem number. These are also the values indicated by the DEFAULT keyword in AUTHORF.

ACCL		5	
C	cards	400	
CM	words	60000B	
CT	minutes	60	
DC	cards	400	
DCM	words	54000B	
DL	pages	40	
DMS	PRUs	0	(unlimited)
EXPDAT		from	application
FILEL	files	15	
L	pages	200	
MAXID	users	3	or from application
MLC		3	
MS	PRUs	0	(unlimited)
PNDBAL	dollars	from	application
PWRQD		OFF	
RG		2	
RUNL	runs	0	(unlimited)
T	seconds	90	

2.7
Accounting

SCOPE/HUSTLER monitors the supplies and system resources used by each job and automatically calculates their dollar value. It records most of this information in the user's dayfile, which is printed with the job output. Complete accounting data is recorded in a system dayfile and periodically copied to magnetic tape. Users are later billed by a program that processes the dayfile tape and totals the charges for each university department and each private account.

The following sections explain how the dollar value of a 6500 job is computed. They also describe the accounting messages printed in the user's dayfile and explain why certain accounting data is omitted. For a description of USERDT, the program that generates monthly billing summaries, execute 'HELP,F*USERDT.'

2.7.1
Calculation of Job Costs

The total dollar value of a 6500 job, denoted by T$, is the product of the rate group factor (RATE) and the sum of eight different computing charges, plus four types of supplies and labor costs.

$$\mathrm{T\$} = \mathrm{RATE}\,(\mathrm{CPU\$}+\mathrm{PP\$}+\mathrm{CM\$}+\mathrm{CR\$}+\mathrm{LP\$}+\mathrm{CP\$}+\mathrm{PT\$}+\mathrm{TP\$})\ +\mathrm{PG\$}+\mathrm{CD\$}+\mathrm{MT\$}+\mathrm{CT\$}$$

The subtotals are for:

Central processor use	$\mathrm{CPU\$} = \mathrm{CPUt} * \mathrm{CPUr}$
Peripheral processor use	$\mathrm{PP\$} = \mathrm{PPt} * \mathrm{PPr}$
Central memory use	$\mathrm{CM\$} = \mathrm{CMt} * \mathrm{CMn} * \mathrm{CMr}$
Computer use in card reading	$\mathrm{CR\$} = \mathrm{CRn} * \mathrm{CRr}$
Computer use in printing	$\mathrm{LP\$} = \mathrm{LPn} * \mathrm{LPr}$
Computer use in punching	$\mathrm{CP\$} = \mathrm{CPn} * \mathrm{CPr}$
Computer use in paper tape reading	$\mathrm{PT\$} = \mathrm{PTn} * \mathrm{PTr}$
Tape reservation	$\mathrm{TP\$} = \mathrm{TPn} * (\mathrm{CPt} + \mathrm{PPt})$
Pages of paper printed	$\mathrm{PT\$} = \mathrm{PGn} * \mathrm{PGr}$
Cards of punched output	$\mathrm{CD\$} = \mathrm{CDn} * \mathrm{CDr}$
Magnetic tape mounts	$\mathrm{MT\$} = \mathrm{MTn} * \mathrm{MTr}$
Connect time (interactive)	$\mathrm{CT\$} = \mathrm{CTt} * \mathrm{CTr}$

In these equations a 't' suffix denotes a time unit, an 'n' suffix denotes a physical unit (such as cards, pages, words, or tapes), and an 'r' suffix denotes a rate in dollars per unit. For example, the central memory charge is equal to the number of words used multiplied by the period of use, multiplied by the dollar rate per word-hour.

Notes:

1) There are two charges involved in printing and punching output. One covers the use of the printer or punch, the other covers the cost of the paper or cards.

2) The peripheral processor time, PPt, is not the actual number of seconds the PPs were in use, but a pseudo-time calculated from the formula,

$$PPT = (1\ ms)PRU + (100\ ms)RA^{1} + CHT$$

where PRU is the number of disk PRUs transferred, RA is the number of RA+1 requests processed, and CHT is magnetic tape channel time. An RA+1 request is a request made by a CPU program to a system PP program, generally for input or output. Channel time is, roughly speaking, the time the tape is in motion.

3) The calculation of CM$ is complicated by the fact that a separate calculation must be made each time the job field length changes. For a given field length, say CMn_i, the dollar value, $CM\$_i$ calculated for that field length is given by

$$CM\$_i = [(CPUt_i - CPUt_{i-1}) + (PPt_i - PPt_{i-1})] * CMn_i * CMr$$

The total central memory dollar value, CM$, is the sum of the $CM\$_i$'s calculated for each field length used by the job.

4) The second factor of the total job cost, the rate group factor, is determined as follows:

For reduced jobs (RG1), RATE=0.5
For normal jobs (RG2), RATE=1.0
For express jobs (RG3) and interactive jobs, RATE=1.5

5) In addition to the dollar value of a job, the user is charged for permanent file storage according to the formula,

$$PF\$ = PFn * PFt * PFr$$

where PFn is the number of disk PRUs occupied by the permanent files and PFt is the length of time they have occupied that space. The permanent file charge is normally calculated four times a day.

6) The rates for each type of service and supply are listed in the *Facilities and Policies Handbook*.

2.7.2
100% Pay vs. Subsidized Accounts

Printed on the right-hand side of the problem number card are the words SUBSIDIZED or 100 PCT PAY. Whether the problem number is billed for all or only part of the calculated job cost is determined by the problem type, which is punched in columns 23 and 24 of the PNC.

Computer service for certain types of projects is subsidized, in part, by the Office of the Provost. Generally, these are projects that receive no support from non-university agencies. Problem numbers assigned problem types 8, 11, 12, 15, 19, 21, and 23, on the other hand, are billed for 100% of the calculated job cost. In addition, off-campus groups (problem type 12) are assessed a 10% surcharge. For additional information about pay policies and problem types, see the *Facilities and Policies Handbook*.

[1] 1ms=1 millisecond=0.001 second.

2.7.3 Dayfile Accounting Messages

The dayfile printed with each job gives a partial account of the total cost for that run. It contains messages giving the central processor, peripheral processor, central memory, tape reservation, card reading, and printing costs:

RATE(CPU$+PP$+CM$+TP$)	RATE(CR$)	(LP$+PG$)
total compute value	card reading	printing

The dayfile does not show charges for tape mounting (MT$), card punching (CP$ and CD$), or permanent file storage (PF$). Also, the dayfile always shows the print charge computed at the normal rate (RATE=1.0), rather than the actual rate. Nor does the print charge message reflect the cost of other printed output created by DISPOSE; each print file contains its own print charge message.

The card reading cost and the "total compute value" are immediately deducted from the PN and ID dollar balances. The printing, punching, and tape mounting costs, on the other hand, are computed from the system dayfile and deducted hourly. Deductions for permanent file storage are usually made four times a day at somewhat irregular intervals. Because of this delay, the user may see sudden drops in the dollar balance, which is printed at the beginning of each dayfile. This dayfile message shows the ID dollar balance before any of the costs associated with the current job are deducted.

Following the dollar balance message is a message giving the number and cost of the cards read for the job, as in the sample below. Unlike the print charge message, this value is computed using the actual rate group factor.

```
001500        CARDS READ     VALUE    $0000001.62
```

The following summary of compute costs is displayed at the end of the user's dayfile.

```
CP USE     10.732 SEC     VALUE$      .45
PP USE     21.705 SEC     VALUE$      .06
CM USE      2.136 W-H     VALUE$      .55[1]
TP RES       .541 MIN     VALUE$      .03
```

The dollar values given in these lines are computed using the normal rate group factor (RATE=1.0). The next dayfile line totals these three amounts and multiplies the sum by the actual rate group factor. For example, if the above lines were issued for a reduced rate job (RG1), the next dayfile line would be

```
TOTAL COMPUTE VALUE AT RG1 $      .55
```

A similar accounting summary is displayed at the end of an interactive session, but includes an additional line giving the connect time charge (CT$). The total compute value for an interactive session includes the connect time cost, but only the CM, CPU, and PP costs are subject to the RG3 premium.

The last line of any output printed at the Computer Center, including print files created with DISPOSE, gives the print cost.

```
000034  PAGES PRINT.    000248 LINES PRINT.  FOR $   000.36 AT RG2
```

[1] The CM USE value, although labeled as word-hours, is actually a unit of block-hours, where a block is 100B (64) central memory words. To get the number of word-hours, multiply the CM USE value by 64.

The page and line count include the dayfile, banner and trailer pages. The dollar value comprises both the computer use value (LP$) and the paper value (PG$). Although the message always shows the print cost computed at the normal rate, the user is later assessed according to the actual rate group of the job. Note that only the lines charge is affected by rate group.

Accounting messages for printing and card reading do not appear on jobs read from or printed at the remote batch terminals since these charges are not assessed at the remote sites.

2.7.4 Other Dayfile Messages

SCOPE/HUSTLER prints a number of dayfile messages that record the use of system resources at intermediate stages of job processing. These include the NL, RP, CPU-PPU, and the FILES messages.

1) After executing each control card, the system issues a dayfile message showing the cumulative central processor and peripheral processor times, as in the following sample.

```
CP-PP SEC.    11.833-    14.844   $   .88
```

Here the PPU time is the actual number of PPU seconds used. The total true PPU time is displayed just before the final accounting summary, e.g.,

```
PP   016.689 SEC.     CHT   12.645 SEC.
```

The actual PP time can be disregarded since the PP charge is calculated from the final RP and CHT values.

2) After each control card that alters the job field length, an NL (new length) message is printed. For example,

```
COMPASS,I=COMPILE.
NL    045000
RP          00000143   000000001700
ASSEMBLY COMPLETE.        42300B SCM USED.
CP-PP SEC.     36.398-    44.894   $   2.73
LGO.
NL    031300
```

This dayfile shows that the job's memory assignment was raised to 45000B to assemble the program on file COMPILE, and then reduced to 31300B after the program was loaded.

3) The RP message, which appears in the example above, is a cumulative record of disk input/output. The first value is a cumulative count (octal) of the RA+1 requests processed by the system, which closely approximates the number of physical read and write requests. The second value is a cumulative count (octal) of the number of disk PRUs transferred. This message is displayed after each field length change. The final RP values are used to compute the PPU charge.

4) The FILES message tells the user the maximum number of files that were assigned to the job at one time, and the maximum number of disk PRUs that they occupied.

```
MAX FILES   0016    MAX PRUS   003700B
```

These values enable the user to specify appropriate file and mass storage limits.

3

Job Structure

Chapter 3 is a discussion of job processing. There are several means of communicating with the computer (e.g. remote batch, interactive, and central site batch). The SCOPE/HUSTLER operating system at MSU accommodates both batch and interactive processing. The differences between these processes will be illustrated in this chapter.

Batch processing typically involves using punched cards as input to the computer. You, the user, are responsible for incorporating information onto cards correctly and making sure that the deck contains a complete job. The information contained in the deck is submitted to the computer in a batch. There is no interaction between you and the computer while the job is being processed. Consequently the information in the deck must be complete and arranged logically. This will be discussed in detail in Section 3.1. After the card deck is read directly into the computer system via the card reader, the data may be stored on magnetic tape or disk. Your output is produced after a varying time period (depending on the system load, time of day, and priority rate). If any corrections are then necessary, you must punch the corrections on new cards, and resubmit the entire deck to the system.

Interactive processing is considerably faster than batch. It provides almost instant feedback with direct interaction between you and the computer. Terminals send your individual instructions to the computer by means of a keyboard, and receive the computer's immediate response to each instruction either on paper (hard-copy terminal) or on a cathode ray tube (CRT terminal).

3.1 Job Structure

Job structure varies depending on whether the job is batch or interactive.

Batch

A batch job can be submitted to the computer system as a deck of punched cards or from storage on disk. In either form a job is considered to be one file, consisting of one or more sections (see Chapter 4 for a discussion of files and sections). Each section is terminated by an **end-of-section** card (EOS), a blank card on which you multipunch a 7, 8, and 9 in column one. The last section of the job deck must be followed by an **end-of-information card** (EOI), on which you multipunch a 6, 7, 8, and 9 in column one. The EOI card must be the last card in your deck, as it tells the computer that this is the end of your file. For more information on the EOS and EOI cards see Section 4.3.1.

Your job deck is automatically read into a local file named INPUT, which then acquires the structure of the job deck. The first section of the deck is always the control section. The first cards in this section are identification and authorization cards—sequence card, problem number card, job card, and password card—while the rest of the section consists of SCOPE/HUSTLER control statements that specify how the job is to be processed. Unlike many operating systems and job control languages, control statements for the SCOPE/HUSTLER operating system appear only in the first section.

A job deck often contains only one section, the control section. If a job deck contains more sections, they are called data sections. A data section may contain one of the following:

a. A source program (any program written in programming language) to be compiled into a binary object deck (written in machine language), which the computer can process.

b. A binary object deck (i.e., a source program which has been compiled).

c. Data for a user or library program.

d. Control directives for a utility program, such as 'UPDATE.', 'PFLOAD.', or 'AUTHORF.'.

Each data section corresponds to a particular control statement which specifies how the data in that section are to be processed. As a consequence of this relationship, the data sections must be arranged in the same order as their respective control statements. The control statements, in turn, must be arranged in the order that you wish to have each operation performed. The following are examples of typical batch job deck structures.

Example 1: Job deck to compile and execute a FORTRAN program.

1st section (control section)	Sequence Card	identification and authorization cards
	PNC	
	Job Card	
	Password Card	
	FTN5.	"compile" control statement
	LGO.	"execute" control statement
	7/8/9	EOS card
2nd section (data section)	Program ABC	source program
	7/8/9	EOS card
3rd section (data section)	data	data to be used in program ABC
	6/7/8/9	EOI card

This job consists of three sections—a control section and two data sections—separated by end-of-section (EOS) cards and followed by an end-of-information (EOI) card. The control section contains the identification and authorization information, and two control statements. The first data section contains the programming language statements of program ABC, and the second data section contains the data to be used in program ABC.

After your identification and authorization information has been verified, the 'FTN5.' control statement is executed. This is the first control statement in the job, and thus controls the corresponding first data section. The 'FTN5.' control statement calls the FORTRAN 5 compiler to compile the source program ABC into machine language (i.e., the compiler will translate program

ABC from the source language FORTRAN, in which it is written, into machine language, called object code). The compiler then writes the translated code onto the file LGO. (LGO is the default name of the file which will contain the object code.)
The second control statement, 'LGO.', instructs the computer to execute the contents of file LGO, in this case program ABC. The data which program ABC requires is found in the data section following program ABC.

Example 2: Job deck composed of only one section.

```
        Sequence Card

        PNC

        Job Card

        Password Card

[1]     ATTACH,PROG,PROGRAMABC.

[2]     FTN5,I=PROG.

[3]     ATTACH,TAPE1,DATAFILE.

[4]     LGO,TAPE1.

        6/7/8/9
```

Notes:

[1] Retrieves program ABC (which resides on disk storage as a permanent file named PROGRAMABC),attaching it as local file PROG.
[2] Compiles program ABC, specifying that the program is found on local file PROG.
[3] Retrieves file DATAFILE from disk storage and attaches it as local file TAPE1.
[4] Executes program ABC, specifying that data is on file TAPE1, to be written on program ABC's file INPUT.

Although this job consists of only one section—the control section—it performs the same functions as the job in Example 1. In this case, however, program ABC and its data are retrieved from permanent files stored on disk rather than contained in data sections of the job deck.

You should not worry if some of the control statements are unfamiliar; this example is intended only to illustrate a job made up entirely of control statements. For an explanation of permanent files and the 'ATTACH.' control statement, see Chapter 5. All control statements are described in Chapter 7.

Example 3 : Job using non-numeric data (a password).

```
Sequence Card

PNC

Job Card

Password Card

AUTHORF,CHANGE,PW.

7/8/9

NEWPASSWRD

6/7/8/9
```

In this job the 'AUTHORF.' control statement changes your password from the old password to the new one, which is presented as data in the data section following the 7/8/9 (EOS) card.

Interactive

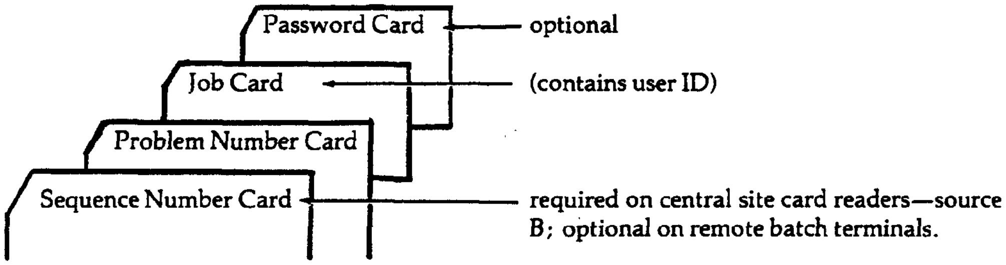
Interactive processing has nothing comparable to deck structure in batch processing. The very nature of the interactive process does not allow a completely prearranged job, but requires that each instruction you give be determined by the computer's response to previous instructions. (You can, however, submit batch jobs from a terminal by using the 'DISPOSE.' statement; see Sections 3.5.1 and 7.4.1 of this manual, as well as Section 3.11 of the *Interactive System User's Guide*.)

3.2 Identification and Authorization

After your application for computer services has been approved and processed, you are assigned a problem number (PN), user name (ID), and password (PW). These are the three items necessary for authorization to use the computer, whether in batch or interactive mode. For full descriptions of the PN, ID and PW, see Sections 2.1.3 and 2.2.

Batch

The first cards or statements of any batch job are identification and authorization statements. The number of these statements required in a batch job depends on (a) where the deck is submitted, and (b) whether a password is required. When present, the four cards must appear in the following order.



The sequence number is used only with decks submitted from the central site card readers (source B). The password is required if the PWRQD flag in the Authorization File is set to ON. (To find out if the PWRQD flag has been set to ON in your authorization file, see Sections 2.6.2—PN and User Fields, and 2.5.4—DISPLAY.)

Interactive

In an interactive job the identification and authorization statements are condensed into one line, as in the following example.

```
password,problemnumber,userid
```

When you activate the terminal, the system prints the following information, and requests your authorization and identification information. **Although a password may be optional in batch mode, it is required in interactive mode.**

```
03/19/81  MSU HUSTLER 2 LSD 50.36  03/16/81  CYBER750

TYPE PASSWORD, PN, AND USER ID.
███████████
```

You must enter the desired information within five minutes or your terminal connection will be disconnected. The system will black out spaces in which to type your password, in order to insure your password's security. If you do not enter the correct information, a message will be printed instructing you to try again.

If, after three attempts, you are still unsuccessful in logging in, the system will disconnect the terminal after displaying the following message:

```
YOU HAVE HAD THREE TRIES. GET HELP.
```

For more details on log-in options in interactive mode, see Chapter 1 of the *Interactive System User's Guide*.

The following pages describe in detail the function and format of each of the authorization statements as they are used in batch and interactive processing.

3.2.1 Sequence Number

Batch

If you enter batch jobs from the central site Input/Output Room (source B), Room 207/208 Computer Center, you must take a pre-punched sequence number card from the rack just inside the door to Room 207. The sequence number card must be the first card of any deck read by the central site card readers. The card contains a sequence number, composed of a two-character source code and a five digit number, which identifies the job within the system and all of its printed, punched, and plotted output. If you enter a batch job from a remote site, your sequence number will be assigned automatically by the computer.

Interactive

Each interactive session is assigned a sequence number by the system and is displayed immediately upon logging in. This number identifies your session and all files disposed to output devices. (This number will be repeated when any file is disposed during the same session.) A new sequence number, different from the session number, will be generated when a file is disposed to the batch input queue. (For more information on 'DISPOSE.', see Chapter 7 of the *Interactive System User's Guide*.)

CAUTION: The second character of the source code changes if output is directed to a site different than the input site. For example, if an interactive job has the number SS00001 and output is directed, via 'DISPOSE.', to source B, the output will be identified as SB00001.

3.2.2 Problem Number

The problem number (PN) is used in both batch and interactive modes as an account number.

Batch

In batch the problem number card (PNC) is a pre-punched card which contains the PN, and is issued by the Computer Laboratory to identify an authorized account. In decks submitted from source B, the PNC follows the sequence card. In decks submitted from remote batch card readers, the PNC is the first card of the deck since no sequence card is required at remote batch sites.

Note that a password card is not required when you dispose a batch job from an interactive terminal. See Section 3.5.1 — DISPOSE.

Columns 1-60 of the PNC contain the problem number, PN expiration date, problem type, and name of the PN holder. Columns 61-80 contain "check digits" which are used to insure that the card is not altered. If a PNC is lost or stolen, the PN Manager can void previously issued cards by having new PNCs punched with different check digits (see Section 2.1.9).

Possession of a PNC is not considered adequate security for your account. The PNC can be easily duplicated on ordinary card stock and your ID is considered public information. PN managers are therefore advised to use the following statement to make password cards mandatory for batch jobs (see Section 2.2).

```
AUTHORF,CHANGE,PWRQD=ON.
```

Interactive

As mentioned in the introduction to this section, the PN is also used as authorization in interactive mode. The function of the PN is the same in batch and interactive modes. The PN is the second item you type when logging in.

password,**problemnumber**,userID

Example:

```
secret,0000001,smith
```

3.2.3 User Identification

Batch

In batch mode, the Job Card specifies the ID which identifies your account. It may also specify limits for a variety of system resources used by the job. If a particular type of limit is not specified, a default value is applied.[1]

The maximum value that may be specified for each type of job limit is determined by your PN limits recorded in the Authorization File. If any of the job card limits exceeds the corresponding PN limit, the job is aborted before it starts executing. During execution, the job is subject to the specified job limits (or their defaults) and will abort if it attempts to exceed any of these. For more information and a list of PN limits and their default values, see Sections 2.6.2, 2.6.3 and 2.6.4.

One purpose of job limits is to trap programming errors that would not otherwise be detected. Unless you know the default limits, know the PN limits, and prepare the job card prudently, however, job limits will abort as many good jobs as bad jobs. To find out what your job limits are, have the computer display your PN limits by using the 'AUTHORF,DISPLAY.' command as explained in Section 2.5.4.

The job card format is shown below. Upper case characters are keyword prefixes (which must be typed as shown, but may be in upper or lower case), and lower case characters represent the values supplied by you, the user. Brackets [] indicate that the enclosed parameter is optional. Optional parameters may be specified in any order.

```
id[,Ccards],[CMwords][,JCcents][,Lpages][,MSprus]
[,MTtapes][,NOINIT|,INIT][,NTtapes][,PNpn][,RGgroup][,Tsec].
```

Parameter	Meaning	Unit	Default
id	the user ID. This is the only required item.		
Ccards	the card limit for punched output. The number you specify is rounded up to the nearest multiple of 8.	cards	DC field from your PN limits.[2]
CMwords	the initial central memory Field Length. For jobs that do not require more than 100000_8words of memory, this value has little meaning. For jobs that require more than 100000_8 words, this value is also called the Maximum Field Length (see Section 7.11).	words (octal)	DCM field from your PN limits.[2]

[1] Your account may actually be part of a larger account assigned to a PN manager. The PN manager's account is identified by the master ID, the ID initially issued with the PN. Any subaccounts subsequently created by the PN manager must be identified by a unique ID.

[2] PN limits for DC, DCM, DL, and DMS fields are described in Sections 2.5.6, 2.6.2 and 2.6.4.

INIT	causes execution of the initialization file, which executes a pre-specified group of statements automatically after logging in. The initialization file can be created and changed only by the PN manager.		
JCcents	the maximum compute cost (total of CP, PP, and CM charges) in cents. This limit cannot exceed either the ID or PN dollar balance. If JC0 is specified, the limit is set to the lesser of the PN and ID dollar balances. As described in Section 3.7, the job cost limit also determines the priority of the job. See also Tsec.	cents	500
Lpages	the page limit for printed output. The number you specify is rounded up to the nearest multiple of 8.	pages	DL field from your PN limits.[2]
MSprus	the mass storage limit (on disk). This limit can be altered during execution via the 'LIMIT.' command.	100_8 PRUs	DMS field from your PN limits.[2]
MTtapes	the maximum number of 7-track tape drives to be used by the job. This number may be altered during execution by use of the 'TAPRES.' and 'RETURN.' control statements (see Sections 6.4.2 and 6.4.3). The maximum number of tape drives available is two.	7-track tape drives	0
NOINIT	requests that normal execution of the initialization file be bypassed (see INIT). This log-in option is illegal if the PN manager has made initialization required.		

[2] PN limits for DC, DCM, DL, and DMS fields are described in Sections 2.5.6, 2.6.2 and 2.6.4.

NTtapes	the maximum number of 9-track tape drives to be used by the job. This number may be altered during execution by 'TAPRES.' and 'RETURN.' control statements (see Sections 6.4.2 and 6.4.3). The maximum number of tape drives available is four.	9-track tape drives	0
PNpn	your problem number. This item is required only for batch jobs created interactively by the 'DISPOSE.' command (see Section 7.5.1).		
RGgroup	the rate group, which determines the job priority and total job cost. See Section 3.7.		2 (for batch jobs)[3]
Tsec	the CPU time limit. If the JC parameter is specified, but T is not, the job time limit is considered infinite.	seconds	7

Examples:

1. JONES.

This job card requests default values for all limits:

RG (rate group)	2 (normal rate)
T (CPU time)	7 seconds
JC (job cost)	$5.00 (500 cents)
CM (max FL)	100000_8 words
tape reservation	0

The print, punch, and mass storage limits are taken from the DL, DC, and DMS fields of your PN limits (see Section 2.6.4).

2. JONES,L250,JC50,RG3.

This job card requests a print limit of 250 pages, a job cost limit of 50 cents, and express priority. Such limits might be appropriate for listing the contents of a file when it is needed quickly.

[3] RG2 is the normal default for batch jobs run between 7:00 a.m. amd 11:00 p.m.; interactive jobs are always run at RG3 at these times. However, both batch and interactive jobs are run at RG2 between 11:00 p.m. and 7:00 a.m.

3. JONES,CM120000,RG1,JC0,T1000.

   This job card specifies a maximum field length of 120000 (octal) words, a time limit of 1000 seconds, and a job cost limit equivalent to the current ID or PN dollar balance, whichever is less. Rate Group 1 is specified in order to reduce total job costs. Such limits might be requested for a very large and expensive job when fast turnaround is not required.

4. JONES,MT1,NT1,JC2500.

   This job card requests a job cost limit of $25, one 7-track tape reservation, and one 9-track tape reservation. Such limits might be appropriate for copying a 7-track tape to a 9-track tape.

Interactive

In interactive mode the ID is typed into the system along with the other two identifiers (password and PN) when logging in. The user ID, password and PN must be correct for you to successfully log in.

3.2.4 Password

Passwords are used to protect individual accounts from unauthorized use.

Batch

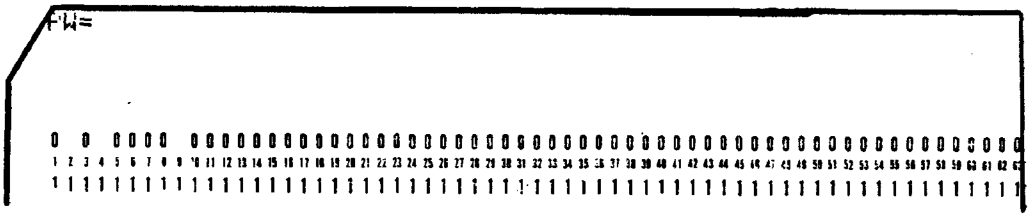
In batch mode the password card is required only if the PN manager has set the PWRQD field of the Authorization File to ON, otherwise the password card is ignored. When present, the password card must follow the job card with this format:

    PW=password

where:

password — is the 1-10 character password that you have chosen (see Section 2.5.6). **Note that there is no terminating period on this card.**

To help protect the secrecy of the password, flip the PRINT switch off when keypunching the password card.



In this example your password has been punched on the card, but is not printed at the top of the card.

Interactive

In interactive mode your password is typed in along with your user ID and PN. Although a password may be optional in batch mode, in interactive mode it is required that you supply all three identifiers to log in. The system will black out the spaces in which you type your entire password and part of your PN. If you want to blank out all three identifiers as they are typed, push the keys "CONTROL" and "V" at the same time. Any characters typed on that line will not print. The "RETURN" key releases "CONTROL-V".

3.3 Control Statements

The first part of this section describes in general terms the syntax of SCOPE/HUSTLER control statements.**Control statement** is a generic term for **control card** (batch use) and **command** (interactive use).

The second part of this section explains the notation used by this and other Computer Laboratory manuals to describe the format of particular control statements. Chapter 7 describes each control statement in detail with examples, while Appendix J gives brief descriptions of all control statements, using the notation of this section. In addition, the *Interactive System User's Guide* gives information on commands available only on the interactive system.

3.3.1 Control Statement Syntax

SCOPE/HUSTLER control statements consist of a **flagword**, followed optionally by a list of **parameters**, followed by a terminator. A flagword is the initial part of a control statement, and identifies the program that will control the processing. A parameter is the part or parts of the control statement following the flagword, which gives the computer specific information about how your input is to be processed. (**Note**: If there exists a local file with the same name as a control statement flagword, that control statement will usually be interpreted as a request to load and execute the local file.) Rules for punctuation and spacing between parameters vary somewhat from one control statement to the next. The following conventions, however, are acceptable to all control statements and should be adhered to unless otherwise noted:

1) Parameters should be separated from the flagword and from one another by commas.

2) Spaces should not be used between or within parameters.

3) The control statement should be terminated by a period.

4) Any characters following an asterisk are treated as comment lines.

5) Characters following the terminator are treated as a comment in batch mode only. In interactive mode characters following a terminator are treated as another command.

Example:

```
ATTACH,PROG,PROGRAMABC.
   ↖        ↖   ↗
flagword    parameters
```

This syntax may be represented by

$flgwrd,p_1,p_2,\ldots,p_k.$

where the flagword is represented by "flgwrd" and the parameter list by "$p_1,p_2,\ldots,p_k$", the last parameter being represented by "p_k". You may insert comments after the period in batch mode only. Replacing the first comma by a left parenthesis and the period by a right parenthesis produces an alternate, universally accepted, format:

$flgwrd(p_1,p_2,\ldots,p_k)$

Example:

```
ATTACH(PROG,PROGRAMABC)
```

You may insert comments after the right parenthesis in batch mode only.

Other Delimiters

In some cases, other punctuation may be used to separate the subfields within a particular parameter. Common parameter forms include:

```
key=value
key=value/value/.../value
key=value=value=...=value
```

Examples:

```
LDSET,STAT=A/B/C,LIB=FORTRAN/CRM.

REQUEST,TAPE,VRN=5001=5002,RW.
```

Literals

Some control statements accept literals. A literal is a character string delimited by dollar signs. With a few exceptions, any parameter field that includes characters other than letters or numbers must be written as a literal. Blanks within the delimiters are retained. If the literal is to contain a dollar sign, two consecutive dollar signs must be typed. For example $XY$$Z$ is interpreted as XY$Z. The descriptions of control statements in Chapters 5 and 7 state whether they accept literals or not.

Example 1:

```
PFLIST,PREFIX=$J.A.L.$,ALL.
```

This program lists all permanent files with names starting with the characters "J.A.L.".

Example 2:

```
PFDUMP,PFN=$MONEY$$PF$,NT=UP1234.
```

This program writes onto tape UP1234 the permanent file named MONEY$PF. (See Sections 5.3 and 5.4.3 for further information on the control statements 'PFLIST.' and 'PFDUMP.'.)

3.3.2
Notation for Control Statement Syntax

Computer Laboratory manuals employ the following notation for describing control statement formats.

UPPER CASE Upper case items are flagwords. These flagwords may be typed in upper or lower case on a keypunch or terminal (unless otherwise stated), but they must be spelled as shown.

lower case Lower case items are to be replaced by you with appropriate symbols or values as defined for that parameter.

[ ] Items in brackets are optional; they may be omitted.

[|] Several items in brackets and separated by a vertical bar (|) are all optional, but one and only one of the items may appear on the control statement. None need be present.

{ | } Several items in braces and separated by a vertical bar (|) represent a list of options of which one must be selected. One and only one of these items must be present.

[...] The preceding item may be optionally repeated on the control statement as desired.

[,...] The preceding item may be repeated, but a comma is required between repetitions.

Items underscored with a single line are assumed as the default condition if no item is specified.

Characters underscored with a double line indicate an abbreviation of the item.

Required parameters, i.e., those that are not surrounded by square brackets, should be specified in the order shown. Optional parameters may be specified in any order provided that a comma is shown inside the left bracket, e.g., [,param].

As an illustration of this notation, consider the 'RETURN.' statement.

RETURN,lfn$_1$[,lfn$_2$][,...][,MT=mt][,NT=nt].

In English, this says that at least one local file name (lfn) must be specified after the flagword: others may follow, each separated by a comma. Tape reservation parameters may be specified, each starting with MT or NT, followed by an equal sign, followed by a number. Some legal forms of the 'RETURN.' statement include:

```
RETURN,A.
RETURN,A,B,C,D,E,F.
RETURN,A,NT=2.
RETURN,A,MT=1,NT=0.
RETURN,A,B,MT=1.
```

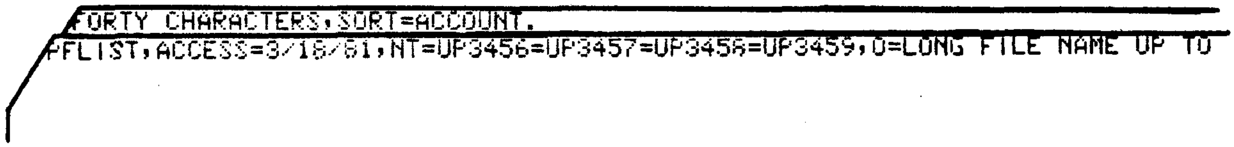
For further information on 'RETURN.' see Section 6.4.3.

3.3.3 Continuation Cards

Batch

A very long control statement may need more than one card to contain it. In such a case, the control statement is continued on a following card (or cards) called a **continuation card**. To signify a continuation card, do not type a control statement terminator (a period or right parenthesis) until the last card. The system examines each control card for a termination mark. When none is encountered, the system assumes that the next control card is a continuation of the first.

Example: The first control card of 'PFLIST.' does not end with a period or right parenthesis; only the second and final card does.



```
FORTY CHARACTERS,SORT=ACCOUNT.
PFLIST,ACCESS=3/18/81,NT=UP3456=UP3457=UP3458=UP3459,O=LONG FILE NAME UP TO
```

Continuation cards can be processed only by these programs:

```
APLIB      CATALOG    FILEDMP    F45       PFLOAD
ATTACH     COBOL      FTN        LISTTY    PFLIST
AUTHORF    COMPASS    FTN5       PFDUMP    REQUEST
BASIC      FILE
```

Lack of a terminator on a control statement which does not require a continuation card can cause serious errors, as in the example below.

```
PNC
.
.
.
REWIND,IFILE
FTN5,I=IFILE.
LGO.
.
.
.
```

Because the terminator is missing on the 'REWIND.' control statement, the 'FTN5.' control statement following it is considered a continuation of 'REWIND.'. The 'REWIND.' control statement does not process continuation cards, so the 'FTN5.' statement (in this example) is skipped. The job will abort because the LGO file is empty, not because of a control statement syntax error; the missing terminator generates an informative dayfile message only.

In programs which do not accept continuation cards, the system will process the first card of the control statement and then, after the program has executed, skip to the start of the next control statement. For more information on programs which allow continuation cards, see Chapter 7.

Interactive

Commands in interactive processing are subject to the same 80-column limit as commands in batch processing. If you need to continue a command on the next line, you will need to create an EXEC file, in which you can continue the command as in batch processing. See Chapter 9 of the *Interactive System User's Guide* for a discussion of EXEC files.

If you are typing a batch program into an interactive file, you may need to continue a line past the margin. The various editing systems available under EDITOR allow line continuation. For more information on EDITOR, see Chapter 3 of the *Interactive System User's Guide*.

3.4 Sample Jobs

This section contains sample batch and interactive jobs.

Batch

Sequence Card	Sequence number card
PNC	problem number card
Job Card.	contains user ID and job limits
PW=verysecret	password
ATTACH,WORK,PWORK.	attaches a permanent file (PWORK) as a local file (WORK)
FTN5,I=WORK.	compiles the attached file into object code
LGO.	writes the object-coded file onto the LGO file, which initiates execution
7/8/9	end-of-section card
data	data used in the program contained in file WORK
6/7/8/9	end-of-information card

The job illustrates the process of attaching a local file and compiling a permanent file. The existing permanent file PWORK is attached and renamed WORK, a local file. The local file WORK is compiled by the FORTRAN 5 compiler. (See Section 3.2 for discussion of the PNC, Job Card and PW.)

Interactive

The following is a sample interactive job.

```
READY 08.24.11
ok
OK-system,fortran
OK-n
100=program tri (input,output)
110=print 300
120=1 x=-1
130=read 100,l1,s1
140=read 100,l2,s2
150=if ((l1.eq.1ht).or.(l2.eq.1ht)) call exit
160=if (l1.eq.1hh) x=sqrt(s1**2-s2**2)
170=if (l2.eq.1hh) x=sqrt(s2**2-s1**2)
```

```
180=if (x.lt.0) x=sqrt(s1**2+s2**2)
190=print 200,x
200=go to 1
210=100 format(a1,1x,f10.0)
220=200 format(e15.5)
230=300 format(* to compute 3rd side of a rt. triangle,
240=+  type: *,/, 5x,*h=n.m for hypotenuse*,/,5x,*s=n.m
250=+  for side*,/,  5x,*t to end program*,/,* where n.m
260=+  specifies length*,*  (format f10.0)*,//)
270=end
280==prompt.
OK-ftner
 COMPILING TRI
    .050 CP SECONDS COMPILATION TIME
 EXEC BEGUN.09.05.23.
TO COMPUTE 3RD SIDE OF A RT. TRIANGLE, TYPE:
    H=N.M     FOR HYPOTENUSE
    S=N.M     FOR SIDE
    T         TO END PROGRAM
WHERE N.M SPECIFIES LENGTH  (FORMAT F10.0)

*h=5.
*s=3.
    .40000E+01
*s=4.
*s=3.
    .50000E+01
*t
*t
  EXIT
    014300  FINAL EXECUTION FL.
      .007 CP SECONDS EXECUTION TIME.
OK-save,intex.
OK-catalog,intex,interex.
 CATALOG,INTEX,INTEREX.
```

In this example, after logging in, you specify the formatting system FORTRAN and initiate automatic numbering. You then type in a FORTRAN program to calculate the hypotenuse of a right triangle. This program is then compiled and executed. When the system prompts for input, you enter the data values. The computer then uses that data to compute the hypotenuse of a right triangle, and prints the output at your terminal. The program is then saved as local file INTEX and cataloged as permanent file INTEREX.

3.5 Job Submission

This section describes the different procedures used to submit batch and interactive jobs.

Batch

Before submitting a job, you should check to make sure that the deck is complete and properly organized. One of the most common sources of error is a hastily assembled control section containing mispunched or misplaced cards. The lister-printer, available in Room 208 Computer Center, can print your card deck on paper, enabling you to scan the contents for errors.

There are several ways of gaining access to the computer in batch mode: via central site card readers, remote batch terminals, and through the Merit Network.

The central site card readers at the Computer Center Input/Output Room (Room 207/208) are available for your use on a self-serve basis during production hours. In addition, you may submit jobs at the Service Window of the Input/Output Room for special handling, e.g., jobs requiring special forms output or jobs to be read at a later time.

In addition to the Input/Output Room facilities, low-speed remote batch terminals are located in numerous sites (other campus buildings and cities). In order to submit jobs from one of these terminals, you must request authorization for that source and approval of the terminal representative by contacting the Computer Laboratory Main Office.

The Merit Network is available to both interactive and batch users wishing access to other computing facilities within the network. The network links the computing systems of MSU, the University of Michigan, Wayne State University, and Western Michigan University. For more details on Merit, its use and rates, see the *Facilities and Policies Handbook*, or refer to the *Merit User Memos*, available in the User Information Center (Room 313 Computer Center).

All output printed or punched at the central site is filed in the Input/Output Room, 208 Computer Center, by the I/O Room staff. Average length printouts are filed in suspended folders according to the last three digits of the job sequence number; large printouts are stacked on a self-service shelf according to the last digit. Large punch output (more than 1,000 cards) is stored in boxes and placed on the same self-service shelf. Smaller decks are stored in a self-service cabinet in the same room according to the last digit of the sequence number. Plot output may be obtained at the Service Window.

Interactive

Interactive sessions with the computer take place on interactive terminals that are hardwired or dial-up. Hardwired terminals are connected directly to the central computer via cables. Dial-up terminals are connected through a normal phone line and a modem (or acoustic coupler) to the central computer system. Public interactive terminals are accessible during published building hours in Room 208 Computer Center. These terminals include hard-copy and CRT models.

Communication between the Cyber 750 and all interactive terminals is handled by the Front-end computer, a Perkin-Elmer 7/32. This mini-computer is connected to the mainframe computer via a high-speed channel interface. The Front-end directly processes control characters which affect terminal function and Front-end commands covering a number of interactive functions. Instructions not concerning terminal function or Front-end commands are transmitted to the operating system of the main computer. (For more information on the Front-end, see Chapter 8 of the *Interactive System User's Guide*.)

As mentioned above, the Merit Network is also available for interactive use. You can access the other Merit sites without going through the local system by using Hermes, a network-to-terminal interface. Telenet, an international telecommunication network, when linked with Merit, allows interactive use of the Merit host computers from anywhere in the United States and a score of foreign countries. For more information on the use of Merit, Hermes and Telenet, see the *Merit User Memos* or the Merit consultant in the User Information Center (Room 313 Computer Center).

Examples of interactive sessions may be found in Appendix H of the *Interactive System User's Guide*.

3.5.1 'DISPOSE' a Batch Job from an Interactive Job

The SCOPE/HUSTLER operating system allows you to type a batch job on an interactive terminal in order to easily edit mistakes. The job can then be sent to the batch input queue from the terminal and run as a batch job. The advantage of this arrangement is twofold: 1) any mistakes in the job can easily be edited while in interactive mode, and 2) you do not have to stay at your terminal while your job is being processed.

The MSU text editing system called EDITOR enables you to build and edit files. For information on using EDITOR, please refer to Chapter 3 of the *Interactive System User's Guide*.

The 'DISPOSE.' command allows you to specify how a local file is to be processed when released from a job. In order to send a file to the central site for batch processing, use this form of the 'DISPOSE.' command:

```
DISPOSE,lfn,IN.
```

The parameter "IN" specifies the batch input queue as the destination. You must be authorized for a particular destination in order to dispose a file to it.

When a job is being disposed to the central site for processing, it must have the same format as a card deck that is submitted at the central site. The file must have all the deck's features except for the sequence card, PNC and password card. The sequence card is not needed since the sequence number is automatically assigned, and the PNC and password card are eliminated because your right to access has been established by successfully logging in. You do, however, need to include the information contained on those cards, i.e., your ID and PN, in this form on the job card:

```
id,PNpn,[,RGrg][,JCct][,CMfl][,Tt][,MTn][,NTx][,Ll][,Cc][,MSm].
```

Example:

```
smith,0000001,rg1,jc2500.
```

See Section 7.2 of the *Interactive System User's Guide* for a description of the job card and its parameters.

CAUTION: Do not make your password part of the job to be disposed to batch, since it is vulnerable to discovery if left in a file.

A 'DISPOSE.' command may be inserted in the control section of the job file to specify the destination, page/card limit, and number of desired copies of the output. If the 'DISPOSE.' command is omitted the output will be printed (or punched) at the central site.

For more information and additional 'DISPOSE.' formats, see Chapter 7 of the *Interactive System User's Guide*.

3.6 Job Processing

Whereas the first part of this chapter described the mechanics of putting a job together, the next few sections explain the logic of the job structure by outlining how the job is processed by SCOPE/HUSTLER.

3.6.1 Input Queue

The input queue holds jobs until they can be processed.

Batch

When a batch job deck is read through the card reader, the system first checks to see whether the required identification and authorization cards are present, correctly sequenced, and properly formatted. If they are not, the job is aborted and an appropriate message is displayed. If they are correct, the job is then copied to disk storage, where it is designated as an input file and identified by the job sequence number. At any given time, there are usually many jobs in this state. Ordered by priority, these files form the input queue. The priority, as computed from the job card parameters Rate Group (RG) and Job Cost limit (JC), is the chief factor determining the order in which jobs leave the input queue and enter the **execution queue** or **pool**. Section 3.7 further elaborates the priority scheme.

Interactive

Interactive jobs bypass the input queue and go directly to the pool.

3.6.2 Pool

From the input queue, the job joins another group known as the pool or execution queue, which is composed of all interactive jobs and up to twenty batch jobs. The pool comprises all jobs eligible for execution at a control point. The pool is a software device that places batch and interactive jobs under the control of a single scheduling algorithm. Internally, the pool consists of tables containing scheduling criteria and other information necessary to interrupt and restart executing jobs.

Oncé the job enters the pool, normal priority (as determined by Rate Group) is ignored. Instead, the scheduler attempts to assign control points to an assortment of jobs which makes best use of central memory, while also providing adequate response to interactive users and distributing access equitably among all jobs in the pool.

Jobs which are in the pool but are not executing (not assigned to a control point) are said to be **swapped out**, which means they have been transferred from central memory to disk and/or ECS (extended core storage). An executing job may be swapped out because

a. it is waiting for the operator to assign a tape or disk file;

b. it is waiting for access to a permanent file presently assigned to another job;

c. it is paused and waiting for a GO command from the operator;

d. it is an interactive job waiting for input from the terminal;

e. it is requesting more central memory than is presently available;

f. the operator has requested a swap out;

g. a higher priority job must be swapped in (e.g., one which would use the resources of the computer more efficiently at the time as determined by the HUSTLER scheduler).

3.6.3 Loading

The Cyber Loader is the system program used by SCOPE/HUSTLER to transfer all object programs from disk storage or magnetic tape into central memory. The loader:

(1) loads absolute and relocatable binary programs,

(2) links separately compiled or assembled programs,

(3) loads subprograms from the system library and links them to user programs,

(4) generates overlays and segments,

(5) prints diagnostics, and

(6) generates memory maps.

Programs that exceed available memory may be organized into relocatable **segments** or absolute **overlays** so that portions or groups of programs may be called, executed and unloaded as needed. Overlay loading is the more efficient technique, because it requires no linking or delinking. However, the linking and delinking overhead of segments is negligible.

A load sequence (load set or load operation) involves all of the loader's processing from the time the loader is called until the time the loaded program is ready to execute.

The type of loading that occurs during execution of the job depends on the size and organization of the job. The two types of loading are (1) core image and (2) relocatable, of which the simpler is core image. Involving only one block of object code (called a core image module), it must be loaded starting at a particular address in memory, which is stored in the header of the core image module. After the loader places the module in memory, execution is initiated at an entry point. The name and address of the entry point are also stored in the module's header. All control-statement-callable system programs, such as 'FTN.', 'UPDATE.', 'ATTACH.', 'COPY.', and 'FILE.', are core image modules.

The second type of loading— relocatable—involves the loading and linking of blocks of object code called relocatable object modules (or subprograms). The relocatable object module is the basic unit produced by a compiler or assembler. It consists of a series of binary tables that describe the subprogram to the loader. Unlike the core image module, a relocatable module lacks a fixed origin address; this is decided by the loader.

For a full description of the loader and commands see the CDC *Loader Reference Manual* or the User's Guide Supplement: *LIBEDIT — Cyber Loader Libraries*.

For more information on pools and control points, see Section 1.4.

3.6.4 Execution

A job can execute only when it is assigned to a control point and given core storage. The control points, like the pool, are simply a software tool for interrelating information about an executing job. Please refer to Section 1.4 for a discussion on control points, control point areas, and the job field length.

Batch

Suppose a batch job is in the pool and is selected for execution by the scheduler. The job is brought to a control point in the following manner.

a. The job is assigned a block of memory (called the job field length) by setting the RA and FL values in the control point area.

b. The job file is designated as a local file and its name is changed to INPUT. (The job sequence number is saved in the control point area.)

c. The first section of INPUT, the control section, is copied to a buffer in the control point area, which also maintains a pointer indicating the next control card to be processed.

d. INPUT is left positioned to the start of the second section (the first data section).

e. A local file named OUTPUT is assigned to the job when needed to collect output for printing.

After the system has completed the job authorization tests described in Section 2.1.6, it begins execution of the control statements, processing them in sequence one statement at a time until:

(1) the last control statement has been processed,

(2) a new control section is read in by means of the EXEC utility (see Section 7.1.2), or

(3) a fatal error occurs.

In both batch and interactive processing, each control statement may be considered a call to load and execute a program. The program referred to by the control statement flagword may be stored either on a system library (as in the case of 'FTN5.'), or on one of the local files assigned to the job (as in the case of 'LGO.').

Note that programs which read from file INPUT read sequentially from a disk copy of the job deck. Therefore, in batch mode, if you want to use the same data section with more than one control statement, you must explicitly backspace INPUT to be positioned at the beginning of that data section. (See Section 7.7.)

Interactive

An interactive job is processed in the following manner.

A. When you log in, the system checks the Authorization File, assigns your job a sequence number, and records the sequence number, ID and PN in the control point area.

B. The system then prompts you for a control statement. Each time you enter a control statement at the terminal, the system:

1) records the control statement in the control point area,

2) swaps the job in to a control point, which assigns that control statement a job field length by recording the address of the job field length (the RA and FL values) in the control point area, and

3) reads and processes the control statement.

Every time you type a control statement, the system reassigns the job field length, depending on the central memory requirements of that particular control statement.

C. When a control statement requires data, the program reads it from the source you have indicated, either from an input file or entered from the terminal. This data is recorded in the job field length.

The distinguishing characteristic of interactive processing is the ability for you to communicate with the computer while it is running your job. The device which makes an interactive conversation with the computer possible is a software item called a connected file. Unlike regular disk files, connected files do not store information; they simply establish a communication path for data flow between your terminal and the computer.

When you are running a program that needs your input, the program will send a prompt to your terminal by issuing a read request on a connected file. When you have finished typing an input line, it is transferred to your job field length. The program then transfers output to your terminal by issuing a write request on a connected file. Once a file is connected, it can be used for input, output, or both; that is, there is no distinction between files connected for read or write operations.

Connected files are used only for communication between you and the program you are running, not between you and the operating system. System commands are read directly by the system, and do not require the use of connected files. Many system commands will, however, connect input/output files for you once they have been called.[4] If you are running a program which does not automatically connect files for you, and you want to have the output displayed at your terminal, you may connect files yourself by using the 'CONNECT.' control statement (see Section 5.2 of the *Interactive System User's Guide*). PASCAL programs are examples of programs which do not automatically connect files. Other such programs are identified in their documentation.

If you do not want to see your output displayed at your terminal, you may disconnect files by using the 'DISCONT.' control statement, or you may designate the output file to be other than the terminal. (An example of this is the control statement 'RANLIB,O=STOREFILE.'. Output from 'RANLIB.' will be listed on the file STOREFILE. The interactive default output file is the connected file ZZZZOT, which is why you would see the output at your terminal if the default were used.) For more information on 'DISCONT.', refer to Section 5.4 of the *Interactive System User's Guide*.

[4] In order to tell if a file is connected or not, type the system command 'FILES.'. Any file name then listed which is preceded by a "C*" is a connected file. An example of this is C*ZZZZOT, where "ZZZZOT" is the file name, and "C*" designates a connected file.

3.6.5 Job Termination

The termination of batch and interactive jobs and the disposition of their files vary depending on certain conditions specified below.

Batch

Upon termination of a batch job, all files assigned to it are disposed of as follows:

a. Permanent files are retained on disk.

b. The job dayfile is written on file OUTPUT; file OUTPUT is changed from type local to type output and renamed to the job sequence number. It joins output files from other jobs that are also waiting to print. These files, ordered by priority, form the print queues.

c. Local files named PUNCH, PUNCHB, PUNCHC, PUNCH9, and P80C are changed to type output and renamed to the job sequence number. These files enter the punch queue.

d. Files disposed to be printed or punched at remote sites are sent to those output queues.

e. Any remaining file with a special (non-zero) disposition code is placed in the print or punch queue, as indicated.

f. Storage held by all other local files is released, and each is evicted from the system.

g. Any tape drives reserved for that job are released and the tape is unloaded.

After all files have been disposed of and final job accounting has been completed, the control point and the job field length are released.

Interactive

There are several ways of terminating an interactive session, depending on how you want your files disposed. If you wish to drop all local files, and do not want detailed accounting information, type the following command:

```
LOGOUT,T.
```

If you want to specify disposition of each local file individually, type:

```
LOGOUT
```

The system will display one local file name (followed by a question mark) at a time. Enter one of the following disposition codes for each local file name displayed.

D to drop the file,

R to retain the file,

T to terminate display of file names and drop all remaining files, or

HELP to request a list of valid responses.

Retained files will be available for two hours after LOGOUT under the user ID/problem number combination which was used to retain them. After you enter a T disposition, or after all local files have been disposed of, the system will disconnect the terminal after displaying accounting information.

For more information on terminating an interactive session, see Section 1.2 of the *Interactive System User's Guide*.

3.6.6 Dayfile

When a job begins execution, the system creates a local file to collect dayfile messages. These messages include accounting and authorization information, error messages (if any), and a copy of each control statement in your program as it is executed. The final lines give a partial summary of the job cost, as described in Section 2.7.3. When the job ends, the dayfile is inserted into file OUTPUT.

The first dayfile line is the system header, which identifies the current operating system. The following example identifies each part of the header.

```
03/19/81  MSU HUSTLER 2     LSD 50.36  03/16/81  CYBER750
```

03/19/81 Today's date.

MSU HUSTLER 2 The name of the operating system.

LSD 50.36 Latest System Delivered number. Revisions made to the MSU operating system, HUSTLER 2, are numbered by the LSD number. There are two kinds of revisions indicated in this number: **major or non-upward compatible**, and **minor** or **upward compatible**. The digits to the right of the decimal point indicate which minor revision is current, while the digits to the left of the decimal point indicate the latest major revision. Minor or upward compatible revisions do not affect a program which has been run successfully before the revision was made. Major or non-upward compatible revisions usually require changes in user programs for them to run successfully. When non-upward compatible revisions are installed, the LSD number is raised to the next integer (e.g., LSD 51.00) and you are notified through various Computer Laboratory publications in advance of its installation.

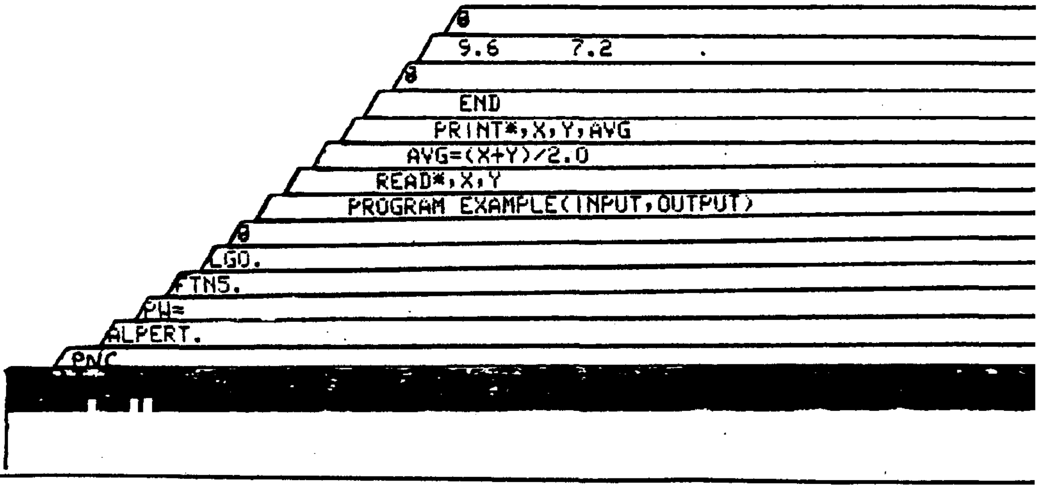
03/16/81 The date that LSD 50.36 was installed.

CYBER 750 The name of the MSU academic computer.

Following the header are several lines of accounting and authorization information. Accounting messages are also interspersed throughout the entire dayfile. In addition, the dayfile includes copies of your control statements, various system statistics, error diagnostics and informative messages.

Batch

The following example contains a batch job card deck and its dayfile.



```
8
9.6      7.2
8
          END
          PRINT*,X,Y,AVG
          AVG=(X+Y)/2.0
          READ*,X,Y
          PROGRAM EXAMPLE(INPUT,OUTPUT)
8
LGO.
FTN5.
PW=
ALPERT.
PNC
```

```
   03/19/81   MSU HUSTLER 2        LSO 50.36   03/16/81   CYBER75
14.14.31.IR3E393
14.14.32.JOB READ-   03/19/81 .14.14.29.
14.14.32.ALPERT.
14.14.32.LAST ACCESS- R 12.14  3/18/81
14.14.32.RUNS- 0029 USER DOLLAR BALANCE $02454.84
14.14.32.J...12      CARDS READ  VALUE  $[illegible].[illegible]
14.14.32.RF  00000000  0000000000000  MAXFL 021100
14.14.32.CP-FF SEC.     .[illegible]-     .[illegible]  $     .[illegible]
14.14.32.FTN5.
14.14.36.         565[illegible] CM STORAGE USED.
14.14.37.         [illegible].326 CP SECONDS COMPILATION TIME.
14.14.37.RP  [illegible]47  [illegible].1616  MAXFL [illegible]555[illegible]
14.14.37.CP-FF SEC.     .044-     4.456  $     .14
14.14.39.LGO.
14.14.41.         END  EXAMPLE
14.14.42.         21000 MAXIMUM EXECUTION FL.
14.14.42.          .314 CP SECONDS EXECUTION TIME.
14.14.42.MAX FILES [illegible]5 MAX PRUS [illegible].
14.14.42.RP  000001[illegible]  000000002155  MAXFL 030000
14.14.42. PF        7.187 SEC.
14.14.42.CP USE       .152 SEC    VALUE$     . .4
14.14.42.PP USE    12.033 SEC    VALUE$     .03
14.14.42.CM USE       .814 K-H    VALUE$     .19
14.14.42.TOTAL COMPUTE VALUE AT RG2 *      .26
14.14.52..[illegible]5 PAGES PRINT. [illegible]38 LINES PRINT. FOR $ [illegible]..7 AT RG2
```

Since dayfile messages are often the only way of detecting errors in your batch program, be sure to check the dayfile of each job you run. See Section 2.7.4 for more information on dayfile messages.

Interactive

In interactive mode the 'DAYFILE.' statement will display specified line ranges of the dayfile at the terminal. Because of the nature of interactive processing, the dayfile is not crucial for detecting errors. As each line of input is entered at the terminal, the computer displays error messages, if necessary.

For more information on the use of the 'DAYFILE.' statement in interactive processing, see Chapter 6 of the *Interactive System User's Guide* (or Section 7.13 of this manual).

3.7 Priority

This section details the software mechanism used to schedule jobs for execution on the MSU computing system.

Batch

When a batch job is submitted to the system, it is placed in the input queue and assigned a base priority. Each job in the input queue periodically has its priority incremented to reflect the fact that it has been waiting to enter the execution queue and should have a higher priority than a job with the same base priority which was just submitted. This process is referred to as **aging**. When a slot is available in the execution queue, the system searches the input queue to find the highest priority job, subject to several **restrictions**, and places that job in the execution queue. These restrictions and base priority are discussed in detail below.

A. **Base Priority**

The base priority of a job is determined by two parameters supplied by the user on the job card. The first of these is the type of service or rate group (RG) in which the job is to be run. This parameter determines the rate to be charged to the user and the **major priority** value. See Table 1 below.

Table 1

Job Card Parameter	Type of Service	Major Priority Value (OCTAL)
RG3	Express	3000
RG2	Normal	2000
RG1	Reduced	1000

Interactive jobs, which automatically run at RG3 rates, will have the highest priority of all jobs in the system. Thus they bypass the input queue and appear directly in the execution queue or pool. Of the batch jobs in the input queue, short Express jobs will have the highest priority, and Long Reduced jobs will have the lowest priority. In general, Express jobs will start executing before Normal jobs. Reduced jobs will not run until 5:00 pm and not until Normal and Express jobs have run. For a given Rate group, Short jobs will start executing before Medium jobs, and Medium jobs will start executing before Long jobs.

The second parameter is job cost (JC). This parameter determines the maximum amount of money which may be used by the job during its execution and also determines the job's minor priority. See Table 2 below.

Table 2

Job Card Parameter	Job Cost	Minor Priority Value (OCTAL)
JC50	0.00 ≤ JC ≤ .50	700
JC500	0.51 ≤ JC ≤ 5.00	500
JC2500	5.01 ≤ JC ≤ 25.00	300
JC10000	25.01 ≤ JC ≤ 100.00	100
JC0	100.01 ≤ JC ≤ no maximum	

The base priority, then, is the sum of the major and minor priority values.

The dollar value of Normal Rate Group jobs will be calculated using the MSU computer service rates. An Express job will cost 50% more than the same job run as a Normal job. A Reduced job will cost 50% of the same job run as a Normal job. Interactive jobs will be assigned a dollar value on the same basis as Express batch jobs (RG3), except between 11:00 p.m. and 7:00 a.m., when they are run as reduced jobs (RG1).

B. **Aging**

Approximately every two minutes, jobs in the input queue have their priorities increased by one. This means that in about four hours a JC500 job would have the same priority as a JC50 job which just entered the queue. Any job with a base priority of 2000_8 or greater may age to a maximum priority of 3777_8. Any job with a base priority of 1776_8 or less may age to a maximum of 1777_8. It takes about 17 hours for an RG2 job of a given type to have the same priority as a similar RG3 job. Note that even though an RG2 job may have an aged priority greater than 3000_8, it still is charged RG2 service rates.

C. **Restrictions**

The system selects a job for entry into the job pool based on priority, and subject to the restrictions listed below.

Note: Certain variables in the restrictions can be controlled by the operator. They are:

*maximum job cost (MAX$)
*small job-cost cut-off (B$)
*number of large job-cost slots in the execution queue (N)
*number of RG1 slots in the execution queue

1. Very large jobs (JC ⩾ MAX$, where MAX$ is normally infinite) will not enter the execution queue. Instead, Operations will make a list of these jobs at the start of each production day and run them serially in order of priority. Only one such job will be allowed in the execution queue at any given time, and none will be started when less than 10 hours of production time remain.

2. Only N large cost jobs (B$ ⩽ JC ⩽ MAX$) are allowed in the execution queue. Normally B$ = $25 and MAX$ is infinite. These values are considered to be RG2, so that if MAX$ = $500, an RG3 job with a JC of up to $750 will be considered less than MAX$. The value of N, B$ and MAX$ can be changed to allow orderly termination of production, to remedy unexpected bottlenecks, and, when necessary, to assure reasonable turnaround time for small batch jobs (JC ⩽ $5) during the prime hours of 8:00 a.m.-8:00 p.m., while servicing large batch jobs during non-prime hours.

3. A maximum of five large CM jobs (CM ⩾ 100000) are allowed in the execution queue, but only two of these may be large JC jobs (B$ ⩽ JC ⩽ MAX$).

4. RG1 jobs are not admitted to the execution queue between 8:00 a.m. and 5:00 p.m. During other hours the number of RG1 jobs in execution will be limited by the operator so as to minimize their effect on RG2 and RG3 jobs. Normally the maximum will be six. There is no guaranteed turnaround time for RG1 and no special provisions will be made to eliminate large back logs in this rate group.

5. No more than 10 tape jobs are allowed in the execution queue. Multi-tape jobs (MT + NT ⩾1) are admitted only when the necessary tape units are available.

Interactive

The concept of "aging" does not affect interactive jobs, since they do not spend any time in the input queue, but go directly to the execution queue. Because of their nature (immediate response from the computer) interactive jobs always have the highest priority rate, RG3, except those beginning between 11:00 pm and 7:00 am. During these hours, interactive jobs are automatically run at RG1 to reduce costs. For details on interactive rates and connect time, see Chapter 1 of the *Interactive System User's Guide*.

4
Files

4.1
File System

Introduction

SCOPE/HUSTLER is a file-oriented system: with few exceptions all information within tl system is organized into files and file subdivisions called sections. Some knowledge of the role ar organization of files is necessary to prepare even a simple job.

This chapter does not describe control statements, utility programs, or other specific file handlir procedures, but explains the concepts, terms, and details of system design needed to understar and apply those procedures. Not all of the information presented in this chapter will be of intere or value to every user. As a minimum, users should learn the terms **local file** and **permanent fil** they should learn the restrictions on file names and the significance of several special file names and they should be aware of the distinctions between the physical and logical structure of files.

Chapter Directory

4.1 **File System**
Introduces the system of data organization used by the SCOPE/HUSTLER operatin system, including the difference between temporary, permanent and local files, the conventions for naming files, and those file names with special significance.

4.2 **Physical File Structure**
Explains how data is physically stored on various storage media; disk, tape, cards.

4.3 **Logical File Structure**
Discusses the logical organization of data used with SCOPE/HUSTLER; defining commor terms and file positions, and going into greater detail on records, sections, section levels and partitions.

4.4 **File Manipulation**
An overview of the file manipulations — reading, writing and positioning — that are possible using SCOPE/HUSTLER control statements.

4.5 **Coded and Binary Mode**
Explains the difference between coded and binary files — the way information is encoded and translated for use by the computer.

4.6 **Input/Output Control**
Describes the user and system tables controlling file processing and non-standard file structures which are available for special purposes. These are esoteric concepts used by relatively few programmers.

For some computer systems, the term "file" refers to a specific type of information stored on a specific device. Under SCOPE/HUSTLER the term "file" applies to virtually any type of information stored outside of central memory. Simply defined, a file is a unit of information which resides on an external device (e.g., tape, disk, or cards) and which is referenced by a file name. Since nearly all information handled by SCOPE/HUSTLER is either a file or part of a file, almost every phase of job processing can be described in terms of files. This is why SCOPE/HUSTLER is said to be "file-oriented."

With input from several batch sites and numerous interactive terminals, there are normally several hundred jobs in the system at one time. Most of these are batch jobs waiting to begin execution or jobs that have already been executed and consist of output waiting to be printed or punched. A job in either of these stages is usually associated with only one set of information.

There are typically about sixty batch and interactive jobs in some phase of execution at the same time. Associated with each of these jobs are various user and system programs, various sets of input data, and various sets of output. All these "sets of information" are files. The task of the file system is to keep track of where these sets of information are stored, to whom they belong, and how they should be handled. This task is accomplished by means of tables stored in central memory, Extended Core Storage and on disk, which define the identity, location, and numerous other attributes of the files. These tables will be described in Section 4.6.

Two major responsibilities of the file system are: to control storage allocation so that two files are not assigned the same location, and to control access so that one job cannot change or destroy a file while another job is trying to read it. To fulfill these duties, SCOPE/HUSTLER requires that:

1. All information used or produced by a job must be defined as a file or part of a file.
2. A job can access only those files that have been assigned to it.
3. Except for permanent files attached with read-only permission, a file can be assigned to only one job at a time. (See Section 5.1.5 for further discussion of multi-read access.)

4.1.1 Temporary Files

Files are said to be **temporary** if they are eliminated from the system when the job is finished. Most temporary files are destroyed as soon as they are returned to the system. That is, if a file is a disk file, the file's allocated storage is freed; if it is a tape file, the tape is rewound and unloaded, and the tape drive is freed.

Temporary disk files may receive special processing before they are destroyed. They may be printed, punched, or placed in the input queue and treated as a separate job: all these are forms of file **disposition.** Certain file names, listed in Section 4.1.5, have a default disposition. Any other temporary disk file can be assigned a special disposition through use of the DISPOSE control statement (see Section 7.5.1).

In interactive use, you may retain temporary files for up to two hours after logging out (by hanging up or using the R parameter on the LOGOUT statement.) Logging back in within this two-hour period using the same ID, automatically assigns the retained files to your terminal.

All files originate as temporary files. Once created, any disk-resident temporary file can be declared **permanent**.

4.1.2 Permanent Files

Permanent files (PFs) are files that may be retained on disk indefinitely. The identification and location of permanent files is recorded independently of other types of files in special disk-resident tables, which are protected from accidental destruction. In addition, you can protect the privacy of permanent files by specifying a set of passwords to control various types of access (See Section 5.1.4).

When a temporary file is made permanent, it is **cataloged** as a permanent file. In order to use a previously cataloged file, you **attach** it as a local file. But unlike temporary local files, the disk space allocated to a permanent file is not released when the file is returned. When an attached permanent file is made temporary again, it is said to have been **purged**; its disk space will be released when the file is returned. See Chapter 5 for a complete description of the permanent file utilities: CATALOG, ATTACH, and PURGE

4.1.3 Local Files

The files assigned to a particular job are said to be **local** to that job. Local files are sometimes said to be **attached** to a job. When no longer assigned or attached, they are said to be **released**, or **returned** from the job. There is a limit to the number of local files which can be assigned to a job at any one time. This is set in the Authorization File. Defaults range from fifteen to sixty-three local files. If you need more files, you can return unneeded files prior to job termination by using the RETURN, DISPOSE, or UNLOAD statement. All remaining local files will be returned at job termination. When a local file is returned, the processing of the file depends in part on whether it is temporary or permanent. Temporary files are eliminated from the system. Permanent files are retained.

4.1.4 File Names

Local File Names

Each file in use by a job is identified by a local file name, which:

1. consists of one to seven alphanumeric characters (A through Z, and 0 through 9),
2. the first of which must be alphabetic (A through Z).
3. Each local file name must be unique within a given job.

Two different jobs may use the same local file name since the system can distinguish these files by other means.

Permanent File Names

Each permanent file is identified by a permanent file name, which:

1. consists of one to forty alphanumeric characters, and
2. must be unique within the system.

If you choose a permanent file name (pfn) that is not unique, the operating system will automatically prefix the pfn with a random digit. you will receive an informative message in your job dayfile when this action is performed. When a permanent file is attached to a job, it has both a local file name and a permanent file name.

4.1.5
Special File Names

Certain local file names have a special significance to the system. Those listed below as **special names** are associated with a special disposition or special processing; that is, they are assigned to user files for a particular purpose. **Reserved names** are for system use and should not be used by other users.

Special Names

INPUT This is the name automatically assigned to the job file of a batch job when it begins execution. In other words, a reference to INPUT is a reference to one or more of the data sections of the job deck.

OUTPUT This is the standard print file, assigned automatically to every batch job. At job termination OUTPUT is printed at the site of job origin, unless you specify otherwise by means of the DISPOSE control statement (see Section 7.4.1).

EWFILE This is the name of the EDITOR work file assigned automatically to an interactive session when an EDITOR directive is issued. All EDITOR directives operate on the contents of EWFILE, which is assumed to consist of numbered text lines written in a special format (see the *Interactive System User's Guide*.)

LGO This is not a special file name in the same sense as the others on this page, but it is commonly used as the default name for object code output by many CDC language processors. Note that the LGO control statement is therefore no different than any other call for loading and executing a user program.

TTYTTY This is an interactive communication file.

INITFIL This is the local file name of the initialization file which may be implemented by the PN manager (See Section 7.1.2).

When returned by a batch job or at the termination of an interactive job, the following files are punched at the central site, unless the user has specified otherwise by means of the DISPOSE statement.

PUNCH The contents of this file are punched in 026 keypunch codes. "Records" (see Section 4.3.1) greater than 80 characters are continued onto successive cards.

PUNCH9 The contents of this file are punched in 029 keypunch codes. Unit records greater than 80 characters are continued onto successive cards.

PUNCHC Like PUNCH, except that the first 80 characters of each record are punched on one card and the excess characters are discarded.

PUNCHB The contents of this file are punched in standard binary format, containing a 7 and 9 multipunched in column one of each card.

P80C The contents of this file are punched in free-field binary format, sixteen central memory words to a card.

Reserved Names

ZZZZxxx The lower case x's indicate that several files have the prefix ZZZZ. Files with reserved names are intended for system use only. These names should not be assigned to a user file. The consequences of a file name conflict between a user file and system file are unpredictable.

4.2 Physical File Structure

Physically, a file is composed of **physical record units**. A physical record unit is the smallest unit of information that can be transferred to or from a file device. The size of a physical record unit (PRU) is determined by the storage medium (i.e., it is device dependent). The specific size and format of a PRU are chosen to take advantage of the mechanical characteristics of the device. On tape we use the term **block** to describe a physical record; on disk the correct term is **physical record unit**. In this chapter, the term physical record unit will be used unless it is necessary to differentiate between disk and tape.

To make an analogy to printed text, a physical record unit is similar to a page in a book. A page can contain information from any part of a chapter: beginning, middle, or end. A page can contain an entire chapter. A page may or may not be full. In other words, the physical structure of the book — the way it is divided·into pages — is independent of the logical structure of the book. This is also true of files.

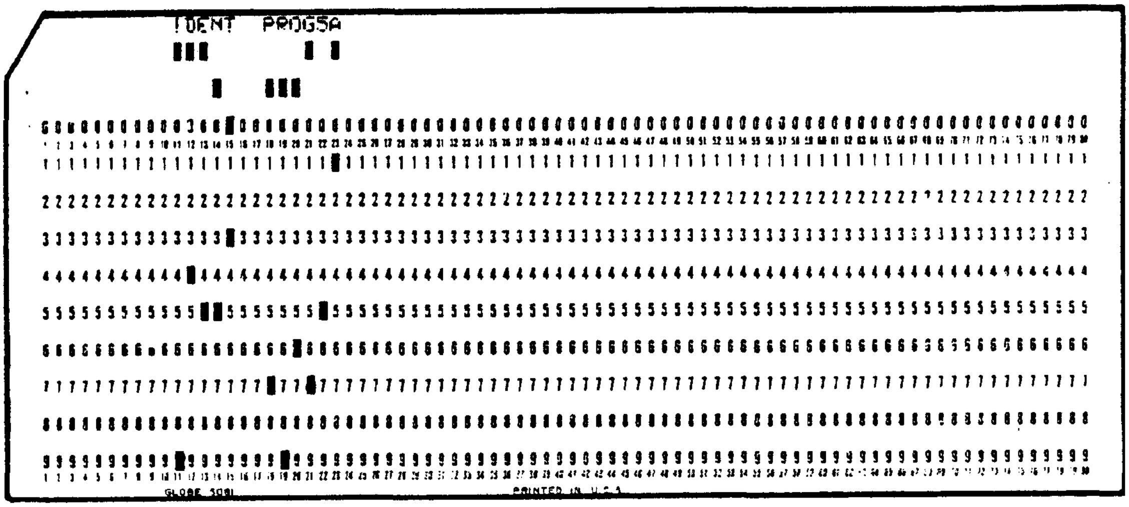
The information stored on disk, tape, and card files is written as a sequence of bits. The term "bit" is derived from "binary digit," i.e., a digit having the value 0 or 1. The physical representation of a bit depends on the device; it may be the presence or absence of a punched hole, the presence or absence of a magnetic flux change, or the positive or negative direction of the flux change. Perhaps the closest analogy for the bit in a printed medium is the character, since it is the smallest physical element of information. But whereas a character may have more than a hundred forms, the bit has only two. Thus, computers adopt coding schemes using 6, 7, or 8 bits to represent each character (see Section 4.5).

When discussing files, we are primarily concerned with units of information stored on magnetic disks and magnetic tapes. Files are also stored on punched cards, but your program never actually reads data from a card file. Instead when your program "reads a card" it reads from a disk file containing a copy of the card images in the deck. Nevertheless, it is appropriate to speak of "card files" since they have the same logical structure as disk and tape files.

Within the storage device, bits are grouped into physical record units in much the same way that characters are grouped into pages. In the computer, a physical record unit is the smallest unit of information that can be read or written on a file device. The device cannot read or write part of a PRU in one operation, and the rest of the PRU in another operation. In terms of our analogy, you cannot turn half a page while reading a book; or print less than a page at a time using most printing processes.

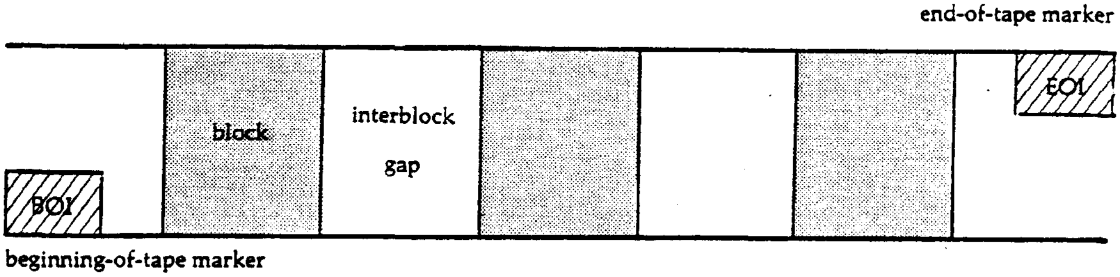
The structure of bits within the physical record unit varies among different devices. On cards, data is punched in eighty 12-bit columns. On magnetic tape, data is recorded in 7-bit or 9-bit columns across the width of the tape. On disks, bits may be written serially on one surface or in parallel on several surfaces and/or several tracks. The arrangement of bits within the physical record unit is generally not important to the user, but it points up problems that the operating system must solve in order to define a logical structure which can be transferred intact from one device to another.

Figure 4-1
Card Example



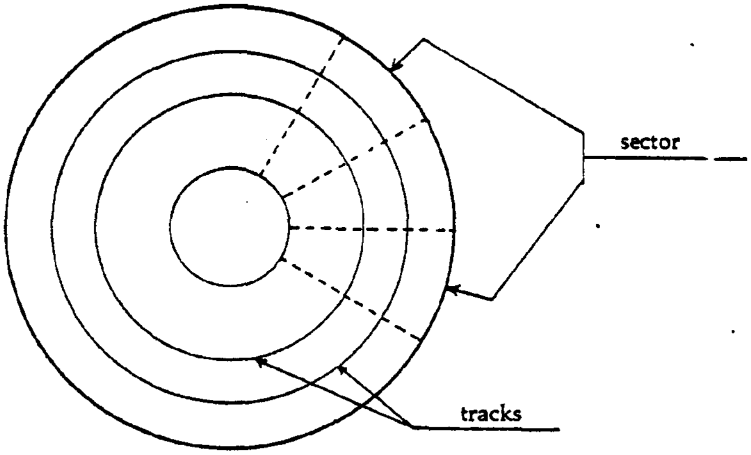
On cards, a single card image is one physical record unit.

Figure 4-2
Tape Example



Tape is a sequential access medium. On tapes, a block is one physical record. Blocks are delimited by interblock gaps. Additional data verification information is included when the tape is written. For more information on tapes, see Chapter 6.

Figure 4-3
Disk Example



Disk is a continuously rotating, random access medium. The concentric circles on the disk surfaces are **tracks**. Each track is divided into the same number of equal length **sectors**. A PRU is stored on one sector.

The length of a PRU may be an unalterable attribute of the device, or it may be determined by the operating system, or it may be left to the user's choice.[1]

Here the term "length" refers to the number of bits or characters of data contained in the physical record unit rather than a dimension of the storage device. The maximum length PRU permitted for a given device, mode, and format will be called the **PRU size** for disk and **block size** for tape. The following sizes are given in terms of 60-bit central memory words.

	PRU size
Disk files	64 CM words
	maximum **block size**
SCOPE coded tapes	128 CM words
SCOPE binary tapes	512 CM words
S tapes	512 CM words
L tapes	unrestricted

[1] SCOPE/HUSTLER permits the user to select the physical record size only for the "stranger" tape file formats (see Section 6.9.3)

4.3 Logical File Structure

Before the days of rotating storage devices, such as magnetic disks, each read or write request from a system or user program called for the transfer of one "record." There was no distinction between a physical and a logical record (what we now call a section). The use of one physical record for each section is feasible for a sequential access device that stops between each operation, but not for a random access device that rotates continuously and must be repositioned for each operation. In order to access data efficiently, the storage space of a disk is divided into fixed-length addressable units — PRUs. If each section were written as one unit, short sections would waste too much space, and long sections would simply be prohibited.

At this time, logical file structure is completely independent of physical file structure. A "record" is a logical construct defined directly by the user (or indirectly through a compiler). You determine the logical relationships between the pieces of information you wish to manipulate. For ease of communication, certain data organization terms[2] are defined here.

record a group of related characters. You (or a compiler) define a record format which specifies how a record begins and ends.

section one or more logically related records. A section begins with the first record after the end of the preceding section and ends with an end-of-section mark.

partition one or more logically related sections. A partition begins with the first section after the end of the preceding partition and ends with an end-of-partition mark.

file a logically connected set of information. Each file has a name. A file begins with the beginning-of-information (BOI) and ends with the end-of-information (EOI).

These logical divisions allow you to structure the information in a file. In terms of our analogy to a book or manual, a record can be seen to be a sentence — a unit of data beginning with a capital letter and terminated with a period. A section could be a chapter. The file could be the book itself. Just as an author can choose an appropriate format for his/her ideas, you as a computer user can choose an appropriate format for your information. Note that these data organization units are hierarchical in structure and differ primarily in how they begin and end.

The following subsections elaborate on the structure introduced here, describing how the logical subdivisions of a file begin and end, the structure of records, sections, and partitions, and the concept of section levels.

[2] The terms used to describe the logical structure of files have changed over the years. You may encounter words used in an outdated sense. "Record" used to refer to both physical record units and sections. Sections were also called "logical records." The term "unit record" was used instead of the current term record. On some occasions the term file referred to a partition, and on other occasions file referred to the entire file. This may explain some apparent inconsistencies—e.g. the system displays EOR instead of EOS for end-of-section and EOF or EOR17 instead of EOP for end-of-partition.

4.3.1 File Positions

A file position is a physical or logical point within the file. Because information is always transferred to and from the file devices in physical record units, the file will always be positioned between two PRUs or blocks. This may or may not coincide with a logical division. When dealing with a higher level language, such as FORTRAN and COBOL, the current file position is masked by the use of buffers and is of no concern to you. Certain file positions coinciding with logical divisions are significant enough to have names: beginning-of-information, end-of-information, end-of-section, and end-of-partition.

File Delimiters: BOI, EOI

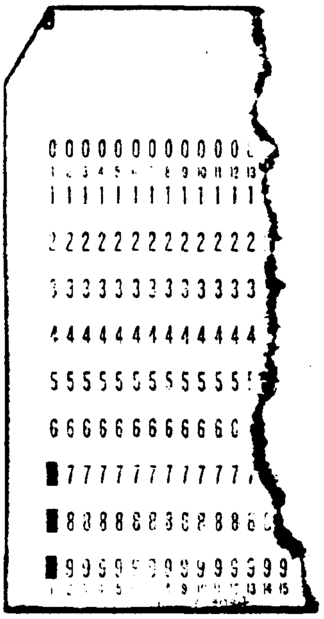
The position preceding the first PRU in a file is called beginning-of-information (BOI), and the position following the last PRU is the end-of-information (EOI). BOI and EOI are the physical beginning and end of the file. You may not read or skip to a point beyond the boundaries of your own file.

Section Delimiter: EOS

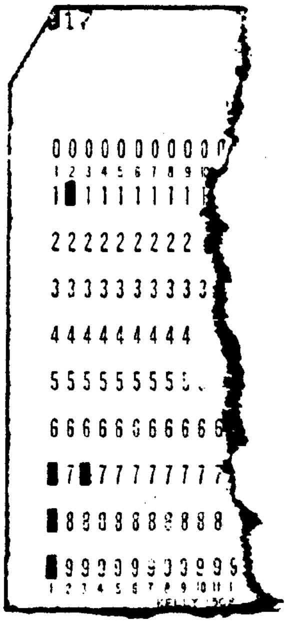
The position following the last record in a section is called the end-of-section (EOS). In a card deck, end-of-section is indicated by a card containing 7, 8 and 9 multipunched in column one. An end-of-section may be followed by a level number (see Section 4.3.4). If the level number is omitted, level 0 is assumed.

Partition Delimiter: EOP

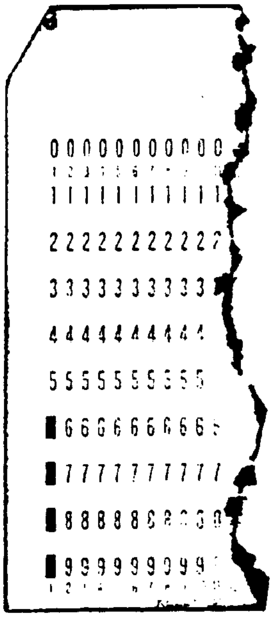
The position following the last section in a partition is called the end-of-partition (EOP). This is equivalent to a level 17_8 EOS (see Section 4.3.4).



EOS card (level 0)



EOP card



EOI card

4.3.2 Records

A record is a group of characters (or bits) that are related to each other by convention. It is the smallest collection of information that is passed between the operating system and you. You (or the compiler) define the structure and characteristics of records within a file by declaring a record format. The beginning and ending points of a record are characteristic of the format.

There are several record formats available. One common type of record is the zero-byte terminated record.[3]

Many input and output devices have media, structure, and coding schemes which are readable by humans. The most common of these are media consisting of paper or cards structured into variable length lines of characters. When a computer system file contains such textual information it implicitly has another logical substructure: it is divided into lines by terminators called end-of-lines (EOLs). Another term for a line within a section is a **record** and the I/O devices (such as card readers and line printers) handling such text files are sometimes called unit record equipment. To be meaningful, text files to be printed contain lines whose maximum length is no greater than the output device on which they are to be written, otherwise a loss of information occurs. Upon input, the trailing blanks following the last non-blank character on the line are removed to save space. A line (record) consists of whole words of ten characters per word. The end-of-line (record marker) must be represented by at least two and up to eleven character codes containing all zero bits (i.e. 2 to 11 display code characters which represent 12 to 66 zero bits).

Figure 4-4

text line	word 1	word 2	word 3
TESTING,TESTING,1,2,3,4	TESTING,TE	STING,1,2,	3,4
(Display Code)	24052324111607562405	23241116075634563556	36563700000000000000
3.14159265358979424	3.14159265	358979424	
(Display Code)	36573437344044354140	36404344424437353700	00000000000000000000

Although both text lines use three words, the second actually contains 19 characters or two words. The third word is necessary to fulfill the end-of-line (EOL) requirement of at least 12 zero bits.

4.3.3 Sections

All files consist of one or more sections, which in turn consist of a sequence of PRUs. Each section starts at the beginning of a separate PRU. Two sections never occupy the same PRU.

Except for magnetic tape files written in the S and L tape formats (see Section 6.9.3), data is always written in multiples of central memory words. Aside from this restriction, a section can be any length. If, for instance, a file is composed of sections that are shorter than the PRU size, each section will be written as one physical record. If a section is longer than the PRU size, all but the last physical record will be maximum length.

To further illustrate the independence of the physical and logical structure of files, consider our analogy between a section and a chapter. Note that each chapter in this manual begins at the top of a page. This practice almost always leaves a partially empty page at the end of each chapter, i.e., a short PRU. (A short PRU is the same length as any other PRU in the file. 'Short' refers to the amount of data in the PRU.)

[3] This discussion and example are adapted from the *University Computer Center User's Manual* from the Universitv of Minnesota.

Suppose, now, that the text of this volume were double-spaced or that it were printed on smaller sheets of paper, representing a change in the PRU size. Although each page would contain less information, the content of the chapter would be the same, and the concepts of full and short PRUs would still apply. In this way, information can be transferred between devices having different physical record sizes without destroying the logical structure of the file.

4.3.4 Section Levels and Partitions

When writing a file, the system automatically appends an eight-character level mark to the last PRU of each section[4]. The level mark, also referred to as an **end-of-section** (EOS) mark, contains a number from 00_8 to 17_8 specifying the rank of the end-of-section. Level numbers are used to group sets of sections within a hierarchy of up to 16 levels. If the section happens to be an exact multiple of the PRU size (so that the last PRU of the record is maximum length), the level mark is written as a separate physical record, which is said to be "zero length" because it contains no data. (Once again the zero-length PRU is the same physical size as all others in the file: it simply contains 'zero' data.)

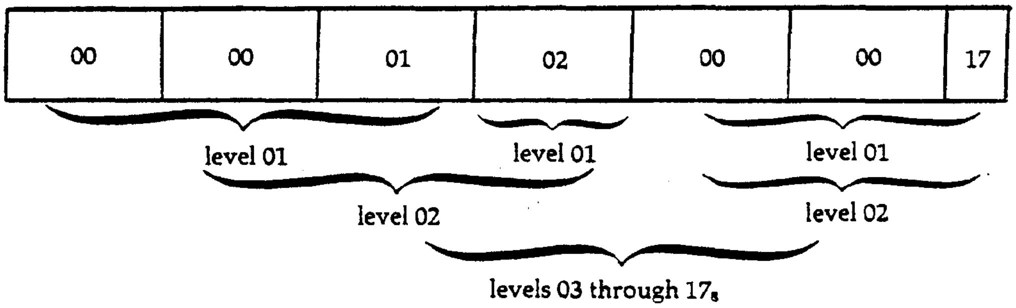
To refer once more to our analogy, section levels are similar to the hierarchy of sections and subsections in this manual. for example, Chapter 4 is divided into six parts labeled 4.1 to 4.6. In turn, 4.1 is divided into five subsections, 4.1.1 to 4.1.5. This is a hierarchical arrangement of information. Similarly, you may define a hierarchy in your file using section levels.

This feature is seldom used, however, because the high-level programming languages—e.g. FORTRAN, BASIC, COBOL—do not provide not provide means for specifying the level number. So, most files consist of one or more level 0 sections (specifying end-of-section) and terminate with a zero length PRU containing a level 17 mark (specifying end-of-partition).

The lowest level number within a file is associated with a single section. A higher level EOS mark defines a set of sections, consisting of the section being terminated and the immediately preceding sections of lower levels. A level 17 mark defines a partition consisting of all preceding sections of level 0 through 16.

The use of level numbers is best explained by example. In the following diagram each block represents a SCOPE section. The number inside each block is the level of the EOS which terminates that section.

Figure 4-5



[4] This does not apply to S and L tapes, but is true for all disk files and SCOPE tapes.

When the user specifies a level 01 EOS for the third section, the first three sections are, in effect, grouped together into one section of level 01. Similarly, the level 02 mark in the fourth section defines a set consisting of four level 0 sections—or two level 01 sections.

Note that a section assigned a level number of n is also considered a section of all levels less than n. For example, if SCOPE/HUSTLER were instructed to skip four sections of level 0, the file would be positioned to the start of the fifth section. If instructed to skip three sections of level 01, the file would be positioned to the end-of-partition.

Level numbers are assigned according to the following rules.

1. If a level number is not specified, as in the case of FORTRAN, BASIC, and COBOL programs, level 0 is assigned to each section.
2. When a level number is specified, as in the COMPASS WRITER macro, the specified level number is written in the EOS mark appended to the section.
3. If the user's buffer is empty when a level number is specified, SCOPE writes a zero length PRU containing the specified level number.
4. When writing an end-of-partition, SCOPE/HUSTLER ensures that the level 17 mark is written as a zero length PRU by first generating a level 0 EOS for the section in progress. End-of-partition is written by ENDFILE statements in FORTRAN and WRITEF macros in COMPASS. The FORTRAN and COBOL input/output routines also write an end-of-partition when an output file is rewound (or simply closed in COBOL) and when the program terminates.

4.4 File Manipulation

The SCOPE/HUSTLER operating system allows you to selectively read, write and position a file. These are all forms of **file manipulation**. All file manipulations deal with logical subdivisions of the file.

4.4.1 Types of File Manipulations

SCOPE/HUSTLER provides several control statements for file manipulation. These are ennumerated and detailed in Chapter 7. Basically, a file is manipulated section by section. A file is **read** until an end-of-section is reached. No explicit control statement is necessary to read a file. To **write** a file requires a control statement which copies sections of one file to another file. This information may or may not be reformatted during the copy operation, at your discretion. The control statement tells SCOPE/HUSTLER how much of the file to copy. See Section 7.5.

After part of a file is read or written, it is positioned at an end-of-section. You can change position within a file by skipping forward using the control statement SKIPF or back using the statements SKIPB and REWIND through the file, but you will always end up at a logical file division (i.e. end-of-section or end-of-information). See Section 7.7.

4.4.2 Rules for File Manipulation

Some general rules for file manipulation are:

1. All read or write operations begin at the current position of a file.
2. Rewind a file before attempting to read from the beginning-of-information.
3. Skip to the end of the file before attempting to write information at the end-of-information.
4. Write operations are usually followed by an end-of-section, then an end-of-information.
5. An error always results from trying to read past end-of-information.

4.5 Coded and Binary Mode

Information undergoes several translations as it is prepared and submitted to the computer for processing. Written characters on coding forms become punched characters on cards, coded characters on disk and binary characters in central memory. The way characters are encoded is called the **mode** of the file. There are two file modes: coded and binary.

Coded files contain data represented according to certain industry standards established to facilitate the exchange of data between computers having different internal coding schemes.

Binary files contain data represented in exactly the same form as it is represented in memory. On disk the distinction between coded and binary mode is irrelevant, since everything is in Display code.

4.5.1 Types of Coded Files

On the MSU computer system there are several types of coded files: ASCII, Display code (or Old Mistic), BCD and EBCDIC.

Interactive terminals transmit (8-bit) ASCII (Americal Standard Code for Information Interchange) characters. There are 128 characters in the full ASCII character set, including: numerals, special characters, both upper and lower case alphabetic characters, and device control characters. Two forms of the ASCII character set exist at MSU, ASCII (AS) and ASCII Fancy (AF). See the *Interactive System User's Guide*, Chapter 5, for details.

(6-bit) Display code (called Old Mistic or OM in interactive use) does not contain lower case letters and lacks some of the special symbols available in full ASCII.

There are also (6-bit) BCD (Binary Coded Decimal) for 7-track tapes and (8-bit) EBCDIC (Extended Binary Coded Decimal Interchange Code) for 9-track tapes.

In addition, cards produced at the keypunch are punched in (12-bit) Hollerith codes, one character per column. Standard 026 and 029 keypunch code conventions reflect industry standards; 026 codes conform to CDC usage, 029 codes conform to IBM usage. All keypunches at MSU use 026 codes. (MSU users can use standard 029 keypunches if they punch '29' in columns 79 and 80 of the jobcard and each end-of-section card in the deck.) When a card file is read into the computer, it is translated into (6-bit) Display code.

Data intended for character manipulation (sorting, etc.) may remain in coded mode. Data intended for numeric calculation must be converted to binary mode: either fixed-point (integer) or floating-point (real) notation. A source program must be translated into CPU instructions in binary machine code (consisting of operation codes, memory addresses, and register ordinals).

4.5.2 Binary Files

Binary representation is an exact image of characters as they would be stored in central memory. Internal representation may vary from installation to installation; it is not subject to industry-wide standards. For example, CDC uses a character represented by 6 bits. IBM, on the other hand, uses 8 bits to represent a character.

On binary-cards, each row position within a column represents a bit. A punch is equivalent to a "1" bit and a non-punch is equivalent to a "0" bit.

Information which is not intended to be printed, or which is frequently used is often kept in its internal form as a binary file. For instance, it is often more efficient to save the binary output file produced by a compiler as a permanent file rather than recompile a program each time it is used.

Example of Data Translation

When the card is read into the computer, the keypunch codes, 1 and 0, are converted to six-bit Display codes, 34_8 and 33_8, for disk storage. (The following example uses octal representation for convenience.)

```
34335555555555555555     Display Code:
                         left-justified,
                         blank-filled. (Blanks are Display code '55'.)
```

Before the 10 can be used as an arithmetic operand, the Display code must be converted to the binary value of 10 in either fixed-point or floating-point format.

```
00000000000000000012     fixed-point

17245000000000000000     floating-point
```

After computation, results and operands must be converted back to Display code characters if they are to be printed.

Alphabetic information is not subject to this type of conversion except in the case of source language programs, where user-intelligible expressions such as

```
IF(X.EQ.0) GO TO 25
```

are translated into their binary machine language equivalents.

4.6 Input/Output Control

The central processor (as described in Chapter 1) communicates only with central memory and ECS, and leaves all input and output tasks to the peripheral processors (PPs). Consequently your CPU program cannot read, write, or act on a file in any way, except by making requests to a system PP program. The execution of these requests involves an interlinked set of user and system tables for identifying and locating the file, for specifying file characteristics and processing options, and for communicating the results of an operation to you.

The manipulation of user tables needed to control file processing can be quite complex. But language processors, applications programs, and utility programs do most of this work, so that few users are ever exposed to the details involved. And as a result, many programmers have only vague notions of how their READ and WRITE statements are executed. The following subsections discuss system and user tables and their function.

4.6.1 Active Files

Local files are just one type of **active** file, which is the term applied to all files currently in use within the system. Active files consist of

local files	assigned to user jobs,
input files	jobs waiting to begin execution,
output files	files waiting to be printed or punched.

Each active file is defined to the system by a **File Name Table** (FNT), which serves as the central link between your program and the system PP routines which transfer data between central memory and the device on which the file is stored. Any file not defined in an FNT is either **inactive** (i.e., an unattached permanent file) or it is nonexistent as far as SCOPE/HUSTLER is concerned.

4.6.2 File Environment Tables and Buffers

Each file which is to be read, written, or otherwise manipulated or positioned by a program, must be provided with a **File Environment Table** (FET) and a **buffer** within your program's allocation of central memory (field length). The FET may be characterized as a parameter table or a communications area through which the program supplies SCOPE/HUSTLER with the information necessary to execute a given file request, and through which the operating system supplies the program with information about the results of the request. If you program in FORTRAN, COBOL, or other high-level languages you are never directly concerned with the FET; it is constructed by the compiler and manipulated by the object-time subroutines of the language, which are loaded with your program. However, if you program in low level assembly language, COMPASS, you have the capability of manipulating the FET.

The standard SCOPE FET is five to thirteen central memory words in length. The FET constructed by a compiler, though, may have additional words. For instance, the FORTRAN compiler creates a 17-word FET for each file named in the PROGRAM statement, and the COBOL compiler creates a 33-word table for each file defined in the File Control paragraph. The additional words contain information used by the language's object-time subroutines and the system I/O routines (CRM, MSURM).

The minimum size of the FET is five central memory words, which allows for processing of any sequential file. Words 6-13 of the FET are needed when blocking and deblocking unit records, processing indexed files, processing magnetic tape labels or processing stranger (S and L) tapes. To give you an idea of its layout, the first five words of the FET are diagrammed in Figure 4-6.

Figure 4-6

	59	47	35	32	23 0
word 1	logical file name				code and status
word 2	device type	flag bits	disposition code		FIRST
word 3	0				IN
word 4	0				OUT
word 5	FNT pointer	record block size	PRU size		LIMIT

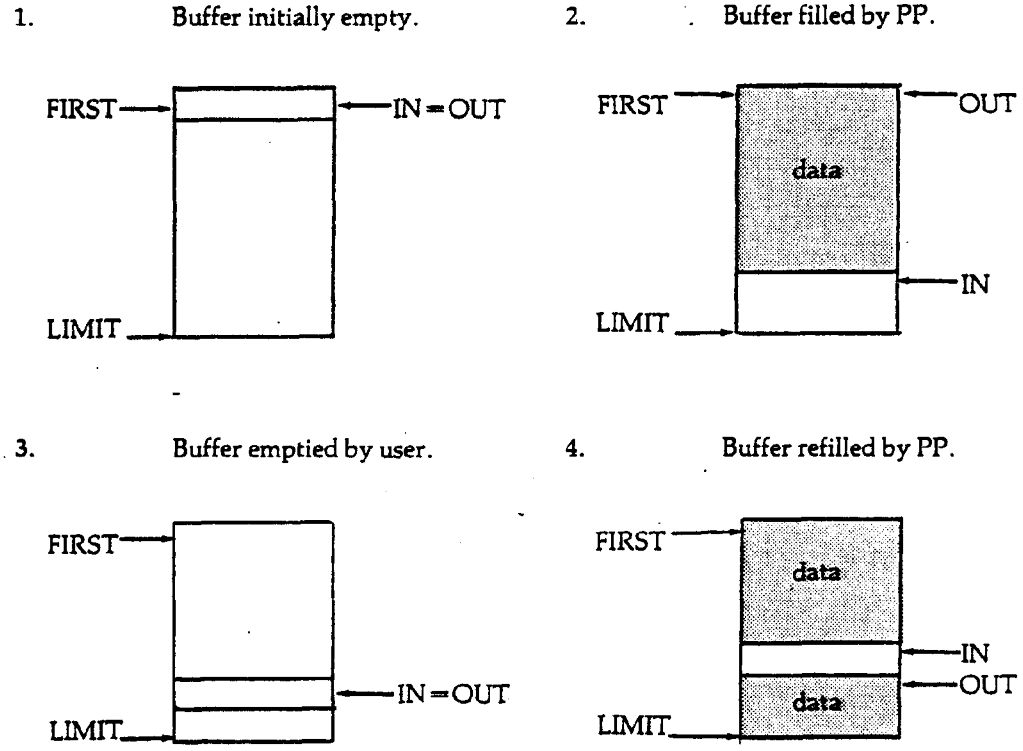
The first word of the FET contains the file name (7 characters, left-justified with zero fill) and an 18-bit "code and status" field, in which the program stores a function code when making a file request and in which the system stores a status code after completing the request. This is how the program is informed of an end-of-section, an end-of-partition, a parity error, an illegal request, etc. Miscellaneous fields in words two and five contain the device type code, the disposition code, the PRU size, and a set of flag bits specifying a variety of processing options. The lower eighteen bits of words 2-5 define the circular buffer.

The buffer is the area of central memory within your field length to which the PP transfers data when reading from the file, and from which it takes data when writing to the file. Since information is always transferred to and from the file in physical record units, the buffer must be at least as large as the PRU size of the file. The size of the buffer is defined by two FET fields, named FIRST and LIMIT, which contain the first word address and the last word address +1 of the buffer. These address fields are called **pointers**.

Two other pointers, IN and OUT, define the location of the data within the buffer. IN points to the location in which the next data word will be entered, and OUT points to the location from which the next data word will be removed. During a read operation, the system advances IN as it fills the buffer, while you advance the pointer OUT as the program removes data for processing. During a write the actions are reversed; you advance IN as you fill the buffer, and the system advances OUT as it transfers the data to the file. The data always occupies the locations from OUT through IN-1, while locations IN through OUT-1 are considered "empty."

The buffer is said to be circular because it is filled and emptied as if it were wrapped around a cylindrical surface, such that LIMIT overlaps FIRST. Data is never stored at location LIMIT, for after IN or OUT advances to LIMIT, it is reset to FIRST. In this manner, OUT follows IN "around the end" of the buffer. Note that when OUT equals IN, the buffer is empty, and when IN equals OUT-1 the buffer is completely full. The buffer is also considered full when the space from IN to OUT-1 is less than a full PRU. The diagrams in Figure 4-7 illustrate the manipulation of IN and OUT during a read operation.

Figure 4-7



1. Buffer initially empty.

2. Buffer filled by PP.

3. Buffer emptied by user.

4. Buffer refilled by PP.

Whenever SCOPE/HUSTLER receives a read request, it fills the buffer until:

1. The next PRU will not fit into the buffer.
2. An end-of-section or end-of-partition is encountered.
3. End-of-information is encountered.
4. A file action error, such as a tape parity error, occurs. (See Section 6.8.7)
5. For S and L tapes, one block is read.

For each write request, SCOPE/HUSTLER transfers as many full PRUs to the file as contained in the buffer. For each write end-of-section or write end-of-partiton request, SCOPE/HUSTLER "flushes" the buffer of all data and writes a short or a zero length PRU, depending on whether the amount of data remaining in the buffer is an exact multiple of the PRU size.

Note that each section requires a separate input or output request. If the section is long, a great deal of data can be transferred with one request. But if the section is short, as little as one PRU can be transferred. The time needed to process a request is on the order of 100 times longer than the time needed to transfer one PRU. A large number of requests for short sections can significantly increase the peripheral processing time in proportion to the amount of data handled. For short sections it is much more efficient to use the central processor to block and deblock zero-byte records than to repeatedly call on the PP routines to transfer data a few words at a time.

4.6.3
File Name Table and File Status Table

A **File Name Table** (FNT) is a three-word system table which is created for each local file, and each file in the input and output queues. The collection of FNTs forms a list of all files known to the system, with the exception of unattached permanent files. When a job is being executed, the FNTs for all of its local files will reside in low core, outside your field length. When the job is swapped out, most (if not all) of its FNTs will be swapped to ECS to conserve central memory.

The first word of an FNT contains the file name, the file type, and the control point number. The second and third words contain different types of information depending on the nature of the file. These last two words form a subset of the FNT, known as the**File Status Table** (FST).

The FNT/FST is the central link between your program and the system PP programs that process file action requests. When your program requests to open a file (an initialization procedure, which is performed implicitly the first time the file is referenced), SCOPE/HUSTLER searches the FNTs for one containing the matching file name and control point number. If the operating system cannot locate the FNT — that is, if the file was not previously defined — SCOPE/HUSTLER will create an FNT and assign a disk unit to the file.

During execution of a request, pointers in the FST are used to locate the file and, if it is disk-resident, to locate the PRU to which it is currently positioned. Other FST fields regulate processing. For example, one field indicates whether the file is open for input or for output. Upon completion of an operation, the system input/output routines store a status code in the FST, which is then passed to the file's FET.

Four file types are defined within the FNT: input, output, punch, and local. A particular file may pass through several of these categories. For example, a job deck begins as an input file but is changed to a local file when selected for execution. And at the end of a job, each local file is either changed to an output file (or possibly an input file), or its FNT is immediately deleted. The transition from one phase of job processing to another consists mainly of rewriting the 3-word FNT.

When a permanent file is attached to a job, SCOPE/HUSTLER creates an FNT specifying type local. The FST entry differs from those of other local files, though, in that it indicates which of the four access permissions — read, modify, extend, and control — have been granted. When the attached permanent file is returned, its FNT is deleted but the file is not destroyed since it is defined independently in special disk-resident tables, called the **Permanent File Directory** (PFD) and the **Record Block Table Catalog** (RBTC).

4.6.4
Record Block Reservation Table and Record Block Table

Recall the organization of disk storage into tracks, sectors, and record blocks. A track is divided into sectors containing the equivalent of 64 central memory words. Disk storage is allocated in groups of sectors called **record blocks**. The purpose of the **Record Block Reservation table** (RBR) and the **Record Block Table** (RBT) is to record which record blocks have been assigned, and to which files.

An RBR is a block of words (in ECS) associated with a particular disk unit. Each bit of the RBR corresponds to a specific record block of that unit. If a bit is zero, the corresponding record block is available and may be assigned to a file; if a bit is one, the corresponding record block is unavailable, i.e., either already assigned to a file or physically unusable.

An RBT is a central memory table (in high core) associated with a particular file. An RBT consist of a "chain" of two-word entries linked together by pointer fields which contain the addresses o the preceding and succeeding entries. The other fields of each entry contain the disk address o each record block assigned to the file. The record block addresses are listed in the order that the are written, thus maintaining the sequential order of the file.

When a disk file is created, SCOPE/HUSTLER assigns a disk unit and then searches the associate RBR for an available record block. Having found one, SCOPE/HUSTLER reserves the recor block by setting the corresponding RBR bit to one the operating system then places the address o the record block in the first entry of the file's RBT, and places the address of the RBT in th FNT/FST. The file will be recorded in consecutive sectors of the record block until it is filled. Then, to write another PRU, SCOPE/HUSTLER must find another available record block and add its address to the RBT chain. As each PRU is written, SCOPE/HUSTLER updates the PRU count in the FNT, plus a pointer to the RBT word containing the address of the current file position.

4.6.5 Rewrite-in-Place

On disk files, all write operations are said to take place at end-of-information because the last link of the record block table chain corresponds to the last PRU written. If a write function is issued when the file is not positioned at EOI, subsequent PRUs are dropped from the record block table chain, and the last physical record written by the operation becomes the new EOI. To replace information in the middle of a disk file, COMPASS provides several **rewrite-in-place** functions, but each record written must be exactly the same length (i.e. contains the same number of PRUs) as the record it replaces.

When a magnetic tape is written, previously recorded data is erased just ahead of the write heads. Still, all write operations define a new end-of-information, just as for disk files. Moreover, there are no rewrite-in-place functions defined for tape files. The reason for this is discussed in Chapter 6.

If you want to be certain of the structure produced when you add information to the end of a file, you must be familiar with the procedures that were used to create the file as well as with those you are using to position the file. If the file ends in an EOP, you can be certain that it is written in a zero length PRU and that the preceding section terminates in a level 0 EOS. By skipping to the end-of-partition and then backspacing over the EOP, you can add another section without starting a new local file. **However, a file does not necessarily terminate with an end-of-partition**. For example, a file written with the SAVE directive in EDITOR will not terminate in an EOP unless the last text line saved is a "*EOF".

4.6.6 Non-Standard File Structures

All I/O operations under the SCOPE/HUSTLER operating system involve files. A record management utility is a set of execution time routines that handles the transfer of data between the user's program and the SCOPE/HUSTLER operating system. When you are programming in a higher level language, many of the details of file processing are built into the language. You need not deal with the details of describing the file because the compiler handles that task automatically.

Record management utilities require the use of PP calls, system tables and system file organization. Individual users can write programs to perform these functions, but most do not, choosing instead to rely on existing system programs. At MSU there are two record management utilities: Cyber Record Manager (CRM) and MSU Record Manager (MSURM). You can call these programs directly using only CPU code (as opposed to PP calls, etc.). Direct calls provide a wider variety of data handling methods and structures than standard I/O statements provide.

Cyber Record Manager

Cyber Record Manager supplies the users with five different file organizations and eight record types. Within these structures most data organization problems can be solved. Data management procedures for higher language programmers are detailed in the appropriate CDC language reference manuals. Background information and operational specifications for COMPASS programmers are fully discussed in the *Cyber Record Manager Reference Manual* and the *Cyber Record Manager User's Guide.* The following discussion outlines the data structures available on the MSU computer system.

File Types

Sequential: records are placed in order of presentation.

Word Addressable: words within the file are numbered 1 to n, each word containing 10 characters. Data is written or retrieved by the word number, which is called the word address.

Indexed Sequential: records are stored in fixed length data blocks using symbolic or numeric keys which are stored in a fixed length index block.

Actual Key: data records of fixed or variable length are stored in fixed length data blocks which contain a fixed number of slots for data. Each record is assigned a key (block and slot number) as it is written.

Direct Access: relative position of records is unimportant. A record is stored and retrieved using a key to generate the address.

Blocking Types

C: character count blocks; produced by FORTRAN Extended formatted WRITE statements. Each block contains a fixed number of characters.

I: internal control word blocks (the first word of every block is reserved by the system for information used internally); produced by FORTRAN Extended 4 unformatted WRITE statements.

E: whole record blocks; frequently found on S and L tapes. The maximum number of characters in the block is preset. This blocking structure is used at non-CDC sites.

Record Types

F: fixed length record.

D: record length is given as a character count, by using a length field contained within the record.

R: a record is terminated by record mark character specified by the user.

T: record composed of a fixed length header followed by a variable number of fixed length trailers; the header contains a trailer count field.

U: record length is defined by the user.

W: record length is contained in a control word prefixed to the record by the record manager.

Z: record terminated by a zero byte.

S: section (in Cyber Record Manager called a logical record). Zero or more PRU's terminated by a short PRU.

MSU Record Manager

MSU Record Manager is a subset of Cyber Record Manager for use with FORTRAN programs. MSURM provides sequential and word-addressable file organizations, C and I blocking types and S, W, and Z type records.

5

Permanent Files

Permanent files are the simplest, most convenient medium for storing information. They relieve the user of carrying burdensome, dog-eared card decks and free him/her of much of the bookkeeping and file manipulation needed to maintain a large number of files on magnetic tape.

A permanent file (PF) is a file stored on disk that can be retained for bye as long as the user likes. The location and identification of permanent files are maintained by the system in special, disk-resident tables known as the PFD (permanent file directory) and RBTC (record block table catalog). These tables ensure that a permanent file is not destroyed when the job using it ends. They also enable permanent files to be preserved during system deadstarts (see Section 1.4.7). In addition, the user may protect the privacy of his/her permanent files by specifying a set of passwords when the files are created.

Any temporary disk file can be made permanent. The action of making a local file permanent is called **cataloging** a file. The file is not transferred or repositioned; the system simply creates an entry in the permanent file tables.

To use a previously cataloged permanent file, the user **attaches** it as a local file. Again, there is no transfer of data from the permanent file to a local file; instead, the system locates the permanent file in the PFD and assigns it a user-specified local file name (i.e., it creates an FNT (file name table) entry). This action establishes the permanent file as an active, local file. When the job terminates, the FNT entry is deleted but, unlike an ordinary local file, the permanent file remains defined in the PFD and RBTC.

SCOPE/HUSTLER provides the following standard permanent file functions.

CATALOG makes an existing temporary file permanent.

ATTACH assigns a requested permanent file to the user's job, subject to permissions tests.

EXTEND makes permanent any information that has been added to the end of an existing permanent file. (The permanent file tables are updated using the local file tables, if appropriate permission has been granted on an attach.)

PURGE changes an attached permanent file to a temporary local file (subject to any permissions which have been granted.)

The control cards that perform these functions are fully described in Section 5.2.

SCOPE/HUSTLER provides several additional utilities that are local to Michigan State University.

PNPURGE enables the user to purge any permanent file charged to his or her account, without having to know its passwords. See Section 5.2.5.

PFLIST reports the status of selected permanent files. The files included in this listing may be selected according to a variety of criteria. See Section 5.3.

PFLOAD reloads specified permanent files from magnetic tape backup copies. See Section 5.4.3.

PFDUMP dumps specified PFs from disk and/or PF dump tapes onto a new dump tape. This allows the user to merge files from disk and existing dump tapes on a new dump tape. See Section 5.4.4.

In addition to PFLOAD, Section 5.4 describes the permanent file backup system and the Computer Laboratory policies regarding permanent file purging, dumping, and reloading.

5.1 Terms and Concepts

Terms and concepts used throughout this chapter are defined below.

5.1.1 Permanent File Names

Within the Permanent File Directory (PFD), each file is uniquely identified by a permanent file name and a cycle number. It is important to understand the distinction between the permanent file name and the local file name. When the user attaches a file, he/she specifies both the permanent file name and the local file name. The permanent file name is used to locate the file in the PFD and assign it to the user's job. After the file is attached, the user always refers to it by the local file name.

Permanent file names may be 1-40 characters, whereas local file names may be only 1-7 alphanumeric characters. The permanent file name may contain non-alphanumeric characters.

5.1.2 Cycles

Up to five files may be cataloged under one permanent file name and its associated set of passwords; each file is called a cycle. Cycles are not necessarily related beyond sharing the same permanent file name and passwords; each is a unique file. Cycles are identified by a number from 1 to 63 assigned by the user when the file is cataloged. If the cycle number is omitted on the CATALOG card, cycle 1 is assumed. Although cycle numbers may be assigned in any order, it is useful to assign them in ascending sequence for the following reason: If the cycle number is omitted on an ATTACH request, the system selects the highest numbered cycle, whether or not it is the most recently cataloged.

5.1.3 WRITE/REWRITE

A write function is one designed to add information to a file (i.e., it writes at the end of the file). A rewrite function is one designed to replace information (i.e., it writes in the "middle" of a file). Under SCOPE/HUSTER almost all output functions, including the FORTRAN, BASIC, and COBOL write statements, are write rather than rewrite functions. When they are used to overwrite the existing contents of a file, all information beyond the newly written data is dropped.

To prevent the user from accidentally destroying a permanent file, the system will permit him/her to write on the file only if it is positioned at end-of-information. The last PRU can be destroyed by a write function. If the last PRU is not a zero length EOF, this can result in a loss of data. If the user tries to write at any other position, the system will abort the job and print the dayfile message,

PF NOT POSITIONED CORRECTLY FOR WRITE

When information is added to a permanent file, the EXTEND function must be executed to make the appended records permanent.

The rewrite functions—namely the COMPASS macros REWRITE, REWRITER, REWRITEF, and WRITIN—are generally difficult to apply. Some system utilities, such as EDITOR, are designed to do rewrites on permanent files. A more typical method of inserting, deleting, or replacing information within a permanent file (or any other type of file) is to generate a second file containing the corrections. This is the method employed by the UPDATE utility.

5.1.4 Privacy Controls

When cataloging a file, the user may specify up to five passwords to protect it from access by other users. A permanent file password consists of 1-9 alphanumeric characters. Each password may control one or more of the following types of access.

Read — Read (RD) permission is required to read the contents of the permanent file.

Extend — Extend (EX) permission is required to write information at the end of the permanent file

Modify — Modify (MD) permission is required to change the existing contents of the permane file, using a rewrite function.

Control — Control (CN) permission is required to purge the permanent file, or to catalog addition cycles.

Turnkey — If a turnkey (TK) password is specified when the file is cataloged, none of the other pe missions can be granted when the file is attached unless the turnkey password is als specified. This amounts to "attach" permission.

To gain a particular type of access permission, the user must specify the associated password on the ATTACH card. If the password does not exist, that is, if it was not specified when the permanent file was cataloged, then that form of access is granted automatically whenever the file is attached. For example, if the CATALOG card specifies only a Read password, then the ATTACH card need not specify any passwords to gain Extend, Modify, and Control permissions. If the CATALOG card specifies a Turnkey password, then the ATTACH must always specify at least the Turnkey password to gain any type of permission.

5.1.5 Multiple-Read Access

The permanent file is the only type of file that can be assigned to two or more jobs simultaneously, provided the following restrictions are met. The system will permit more than one job to read the same file; but the system cannot allow one job to alter the file while another job is attempting to read it. Several jobs can read the same file only if the ATTACH does not give Modify, Extend or Control permission. This is done by either (a) attaching the file with the MR=1 parameter (see Section 5.2.1) or (b) cataloging the file with EX, MD and CN passwords and not supplying any of these passwords on the ATTACH card.

If several jobs attempt to access the same permanent file without multiple-read access, only one job at a time will run. The remaining jobs will be swapped out. These jobs may be rerun or dropped at the operator's discretion. Jobs are not likely to be dropped unless the RERUN parameter is OFF.

5.1.6 Purging/Retention

A permanent file will be retained until it is purged. If the user does not purge his/her own files, they will be purged by the Operations staff when the retention period of the files expires, when the user's PN expires, or when either the user's PN or ID dollar balance is reduced to zero or less. The retention period, from zero days to infinity, is specified on the CATALOG card. If omitted, the default retention period is 15 days.

Before a file is purged by the Computer Laboratory, it is copied to a magnetic tape. If the purged file is still needed, it can be retrieved from this tape within a 14 day period. See Section 5.4 for details.

5.2 PF Utilities

The CATALOG, ATTACH, EXTEND, PURGE, and PNPURGE control cards are described in the following sections.

5.2.1 CATALOG

The CATALOG card catalogs (makes permanent) the local file 'lfn' by assigning it the permanent file name 'pfn', plus optional passwords and identifiers.

```
CATALOG,lfn,pfn[,RP=rp][,RD=rdpw][,EX=expw][,MD=mdpw]
        [,CN=cnpw][,TK=tkpw][,PW=cnpw[,tkpw]] [,CY=cy][,ID=name]
        [,BC=n ][,MR=n].[1]
```

lfn the local file name of the file being cataloged.

pfn the permanent file name which is to be assigned to the file.

RP=rp specifies the retention period (rp) in terms of days ($0 \leq rp \leq 999$). If rp is 0 the file expires at the end of the day. If rp is 999 the file has an infinite retention period. The default is 15 days.

RD=rdpw specifies a Read password, which will be required for read permission whenever the file is attached.

EX=expw specifies an Extend password, which will be required for permission to write at the end of the file whenever it is attached.

MD=mdpw specifies a Modify password, which will be required for rewrite permission whenever the file is attached.

[1] The parameter list of the CATALOG card may be split after any comma and continued on the next card.

CN=cnpw specifies a Control password, which will be required for purge permission whenever the file is attached.

TK=tkpw specifies a Turnkey password, which will be required for all attach requests.

PW=cnpw,tkpw Specifies passwords necessary to catalog additional cycles of a permanent file name. If control and/or turnkey passwords were specified on previous cycles, they must appear here. Up to two passwords may be necessary.

CY=cy specifies the cycle number ($1 \leq cy \leq 63$). If omitted, cycle 1 is assumed. The user may catalog up to five cycles under one permanent file name.

ID=name specifies a 1-9 character owner identifier to aid the user in identifying his/her permanent files in a PFLIST report (see Section 5.3).

BC=n If BC=1, the Computer Laboratory will maintain a copy of the file on a backup tape. If BC is omitted, BC=1 is assumed. If BC=0, the Computer Laboratory will not maintain a backup copy of the file.

MR=n If n is not 0, the file will have Read only permission after the CATALOG is completed, and will be accessible to other users. If n is 0 or MR is not specified, the permanent file will remain attached with all permissions and must be returned and attached with the proper permissions to obtain Read only access and allow access by other users.

All passwords (pw) consist of 1-9 alphanumeric characters. A particular password may be used for several permission types. If the CATALOG does not specify a password for a certain type of access, permission for that form of access is automatically granted whenever the file is attached, unless the multi-read option is in effect.

Since cataloging only involves the manipulation of system tables, it alters neither the contents nor the position of the local file. After the file has been cataloged, it remains assigned to the job. If it was cataloged with MR=1, multi-read access is allowed. If MR=0 or MR is not specified the file is assigned to the job as if it were attached with all passwords specified.

The permanent file name (pfn) consists of 1-40 alphanumeric characters and should uniquely identify the cataloged file unless a new cycle of an existing permanent file name is being created. To ensure that every file is cataloged with a unique permanent file name and cycle number, SCOPE/HUSTLER will add a digit prefix to the user-specified permanent file name in any of the following circumstances.

1. The permanent file name, pfn, already exists and a CY=cy parameter is not specified.

2. The CY=cy parameter specifies an existing cycle of the permanent file name.

3. There are already five cycles of the permanent file name.

4. The PW=cnpw,tkpw parameter did not specify the correct turnkey and/or control passwords during a new-cycle catalog.

Example 1: Cataloging a new permanent file.

```
PNC
JONES,JC80,T10,RG3.
FTN.
CATALOG,LGO,JONESLGO,RD=READ,EX=CHANGE,MD=ALTER,CN=CHANGE,
ID=JONES.
LGO.
7/8/9
     FORTRAN program
7/8/9
     data
6/7/8/9
```

In this example, John Jones catalogs his load-and-go file, LGO, as JONESLGO. When he attaches JONESLGO in a future job, John must specify the password READ in order to load and execute the program. Note that Extend and Control permissions have the same password, and that Modify permission is always granted. Because the CY parameter is omitted, cycle 1 is assumed. Because the RP parameter is omitted, a 15 day retention period is assumed.

Example 2: Cataloging a new cycle of an existing file.

Suppose Example 1 was run earlier and John Jones now catalogs a new cycle of JONESLGO. Because the CN password is defined for JONESLGO, John must submit the password CHANGE with the PW parameter. Cycle 10 of JONESLGO will have the same passwords as cycle 1, but it will have an infinite retention period.

```
PNC
JONES,JC75,T10.
FTN.
CATALOG,LGO,JONESLGO,CY=10,PW=CHANGE,RP=999.
LGO.
7/8/9
     FORTRAN program
7/8/9
     data
6/7/8/9
```

Example 3: A duplicate permanent file name.

Suppose that Sue Jones, unaware that John Jones has already cataloged a file named JONESLGO, submits the following job:

```
PNC
JONES,JC100.
COMPASS.
CATALOG,LGO,JONESLGO,ID=SUEJONES,TK=SENOJ.
LGO.
7/8/9
     COMPASS program
7/8/9
     data
6/7/8/9
```

If everything else is correct, Sue Jones' job will run smoothly. But if she checks her dayfile she will find a message that might read,

```
PF ALREADY EXISTS
FILE CATALOGED AS- 5JONESLGO
```

If she does not check the dayfile she will discover the alteration the first time she attempts to attach JONESLGO. In effect she will attempt to attach John Jones' file using the password SENOJ, which will cause her job to abort with the message,

```
NO CORRECT PASSWORDS SUBMITTED
```

This message will probably puzzle Sue until she realizes that her permanent file name was modified to avoid duplicating an existing name. If Sue has discarded the output from her first run, how can she determine what prefix her permanent file name was given? Solution: She can execute a PFLIST(PNORD) control card, which will list all active permanent files cataloged with her problem number and user ID.

Although there is a simple solution to Sue's problem, she will still have wasted a run. The moral of Sue's story is 1) try to specify a permanent file name that is likely to be unique, and 2)**check the dayfile of every job!**

5.2.2 ATTACH

The ATTACH card is used to access an existing permanent file. It makes the permanent file 'pfn' local by assigning it the local file name 'lfn'.

```
ATTACH,lfn,pfn[,PW=pw1,pw2,...,pw5][,CY=cy][,MR=n].[2]
```

lfn the local (logical) file name that will be assigned to the permanent file. If a file named lfn is already assigned to the job, the job will abort.

pfn the permanent file name of the file to be attached.

PW=pw1
.
.
.
pw5 specifies up to five passwords for any combination of permission types. The passwords (pw1, pw2, ..., pw5) can be listed in any order. Because permission is automatically granted for any type of access which was not given a password when the file was cataloged, the PW parameter may be omitted when no passwords exist for the desired types of access.

CY=cy specifies the cycle number of the permanent file. If this parameter is omitted, the highest cycle number of pfn will be attached.

MR=n If n is 0, the parameter is ignored. If n is not 0, multi-read access is allowed. The file will have only Read permission, assuming the proper RD password was also submitted. No one will be allowed to alter the file at this time, even if they have submitted

[2] The parameter list of the ATTACH card can be split after any comma and continued on the next card.

the proper passwords. Jobs which may alter the file must wait until the multi-read job(s) are completed.

A permanent file may be assigned with Read only permission to several users simultaneously, provided that none of them has Control, Modify, or Extend permission. This entails the specification of CN, MD, and EX passwords when the file was cataloged or the specification of a non-zero MR parameter when the file is attached. If a job attempts to attach a permanent file which is already assigned to another job and multiple access is not possible, the second job will wait until the first has returned the file. Similarly, if a job attempts to alter a file already attached for multiple read access (i.e. the job specified MR=0), the second job will automatically wait for the preceding Read only jobs to return the file. Note: Only batch jobs will wait. Interactive jobs will abort.

Example 1: Attaching a file with Read permission.

Suppose John Jones (from Examples 1 and 2 of the preceding section) wants to execute JONESLGO with a new set of data. He would submit the following job.

```
PNC
JONES,JC40,T5.
ATTACH,X,JONESLGO,PW=READ.
X.
7/8/9
     data
6/7/8/9
```

Because the CY parameter is omitted, the highest cycle of JONESLGO—cycle 10—is attached. In this case, JONESLGO is attached with Read permission only. Extend, Modify and Change permissions were specified on the CATALOG card.

Example 2: An ATTACH involving the Turnkey password.

Suppose a file is cataloged with,

```
CATALOG,TAPE2,PFEXAMPLE,TK=A,RD=B,CN=C,EX=D,MD=E.
```

Consider then the following ATTACH cards:

```
a.   ATTACH,ABC,PFEXAMPLE,PW=A,B,C,D,E.
b.   ATTACH,ABC,PFEXAMPLE,PW=A.
c.   ATTACH,ABC,PFEXAMPLE,PW=B,C.
```

Card (a) will give the user access to PFEXAMPLE with all permission types granted. Card (b) will not allow the user to use the file because none of the other four permissions were specified. The user will receive the message "NO CORRECT PASSWORDS SUBMITTED." Card (c) will cause the job to abort since the Turnkey password must be specified on the ATTACH card in order to gain any of the other four permissions.

Example 3: Multi-read access.

Assume that a file has been cataloged with,

```
CATALOG,FILE,MULTIREAD,CN=C,EX=E,MD=M.
```

The following ATTACH card could be used by two or more jobs to read MULTIREAD simultaneously.

```
ATTACH,lfn,MULTIREAD.
```

Each user may specify a different local file name (lfn), but each must attach MULTIREAD with Read permission only. In this example, none of the users need be concerned with passwords because Read permission is granted automatically.

If the following ATTACH card, which supplies the control password, has been processed:

```
ATTACH,X,MULTIREAD,PW=C.
```

other jobs attempting to attach MULTIREAD will have to wait until this job has returned it.

Suppose a file was cataloged without specifying all Extend, Modify and Control passwords.

```
CATALOG,FILE,PFEXAMPLE,CN=C,EX=E,RD=F.
```

Consider the following ATTACH cards

a. ATTACH,X,PFEXAMPLE,PW=F.
b. ATTACH,Y,PFEXAMPLE, MR=1,PW=F.
c. ATTACH,Z,PFEXAMPLE,MR=1.

Card (a) will not allow multi-read access because Modify permission is granted. Card (b) will allow two or more users to read the file simultaneously. Card (c) will not allow the user to read the file because the Read password is not specified. The MR parameter does not grant Read permission, it merely limits access to the file to Read-only jobs. The user will receive the message "NO CORRECT PASSWORDS SUBMITTED."

5.2.3 PURGE

The PURGE card changes the attached permanent file, lfn, to a temporary local file.

```
PURGE,lfn.
```

lfn the local file name of an attached permanent file.

In order to execute the PURGE function, the permanent file must be attached with Control permission. After the file has been purged, it remains assigned to the job as a temporary file. Like any temporary file, it will be evicted from the system when the job terminates, or when the file is released from the job with a 'RETURN,lfn.' control card.

Caution: One crucial difference remains between a purged permanent file and an ordinary local file. Unless the purged PF was attached with all permissions, it cannot be recataloged. Otherwise, Control permission would be equivalent to all permissions, since then a user could attach a file with Control permission only, purge the file, and recatalog it with new passwords. Thus, the permissions granted when a permanent file is attached preceding a purge remain in effect for the corresponding local file.

Example 1: Getting rid of an unwanted permanent file.

Cycle 1 of MYPF is an obsolete permanent file costing several dollars per day for storage. Assume it was cataloged with,

```
CATALOG,Z,MYPF,RD=VERY,EX=SECRET,CN=STUFF,RP=999.
```

The following sequence of control cards attaches MYPF, purges it, and then releases it from the job, thereby evicting it from the system. (Because the user intends to destroy MYPF, the control password, STUFF, is sufficient.)

```
PNC
Job Card
ATTACH,A,MYPF,CY=1,PW=STUFF.
PURGE,A.
RETURN,A.
6/7/8/9
```

Example 2: An incorrect attempt to purge and recatalog a PF.

In this example, cycle 5 of MYPF is a valuable permanent file that the user wants to retain but for which he/she has neglected to specify an adequate retention period. To avoid having the Computer Laboratory purge the file when it expires, the user decides to attach it, purge it, and recatalog it with a longer retention period. The following deck is submitted:

```
PNC
Job Card
ATTACH,A,MYPF,CY=5,PW=STUFF.
PURGE,A.
CATALOG,A,MYPF,CY=5,PW=STUFF,RP=999.
6/7/8/9
```

Imagine this user's horror when he or she sees a DMPX and reads the fatal error diagnostic in the job dayfile,

```
ATTEMPT TO CATALOG AN EXISTING PERMANENT FILE
```

The user forgot to attach cycle 5 of MYPF with all permissions. After MYPF was purged it was evicted from the system when the job was aborted for the illegal catalog attempt. Fortunately all is not lost. If cycle 5 of MYPF is more than a day old, there should be a backup copy on a permanent file dump tape maintained by the Computer Laboratory (unless the file was cataloged with a BC=0 parameter, in which case, all may well be lost.) Section 5.4.3 describes the control card used to reload permanent files from the dump tapes.

5.2.4 EXTEND

The EXTEND card makes permanent any information that has been written to the end of an attached permanent file.

```
EXTEND,lfn.
```

lfn the local file name of the attached permanent file.

A permanent file can be changed in two ways:

1. Information can be replaced without changing the length of the file.

2. Information can be added to the end of the file, extending its length.

The first case is known as modifying or rewriting the file. The second case is known as extending the file.[3]

Under Case 2, the information appended to a permanent file is not permanent until an EXTEND is performed. If the job terminates without executing the EXTEND card, the appended information is evicted—just as in the case of a temporary file—and the permanent file remains the original length.

The typical user modifies a file (Case 1) only when editing an EWFILE under EDITOR. Because EDITOR not only rewrites records but also adds them, a permanent EWFILE should be attached with both Modify and Extend permissions. EDITOR performs the EXTEND function automatically for the user. There is no direct way to modify a file using only control cards.

Example 1: Copying a new logical record to the end of a PF.

The following job copies data from the job deck to the end of a permanent file, adding a new logical record. This record is then cataloged with the EXTEND card.

```
PNC
Job Card
ATTACH,OLD,FOXYDATA,PW=READ,EXTEND.
SKIPF,OLD,1,17.
SKIPB,OLD,1.
COPYCF,INPUT,OLD.
EXTEND,OLD.
7/8/9
    data
6/7/8/9
```

Read permission is required to skip forward on OLD and extend permission is required for copying INPUT to OLD.

NOTE: This job will only work if OLD ends with an end-of-file mark. Recall from Section 4.2.5 that when SCOPE/HUSTLER writes an EOF, it first terminates the logical record in progress (if any) with a level 0 EOR and then writes the level 17 mark (EOF) as a zero length PRU. The job in this example skips to end-of-file and then backspaces over the zero length EOF mark, leaving the file positioned between the level 0 EOR and the EOF. The COPYCF card overwrites the EOF.

For this particular job, errors would result in the following circumstances.

1. If OLD does not end in an EOF, the SKIPB card will backspace over the last logical record of data. If this record is not a full PRU, it will be overwritten; if it is greater than a PRU, the job will abort with the message "PF NOT POSITIONED CORRECTLY FOR WRITE."

[3] Section 5.1.4 noted that SCOPE/HUSTLER does not permit the user to "write" on a permanent file unless it is positioned at end-of-information. More precisely, SCOPE/HUSTLER permits write functions to overwrite only the last PRU of a permanent file. This allows the end-of-file mark to be overwritten when the file is extended.

2. If OLD contains more than one EOF mark, it will not be positioned correctly after the 'SKIPF,OLD,1,17.' card is executed, and the job will abort.

3. If the information added to OLD is not in the last logical record of the job deck, the 'COPYCF,INPUT,OLD.' card will copy succeeding data decks as well as the intended data deck.

This example illustrates that writing an extension to a file is not always as simple as it might seem. Although it is fairly easy to position a file at end-of-information and then copy some data to it, the resulting file structure depends on the operations used to write the extension and the operations originally used to create the file. Getting desired results requires a fairly sophisticated understanding of file structure and the file manipulation procedures.

Example 2: Recataloging a permanent file.

To avoid creating a new SCOPE logical record each time data is added to the end of a permanent file, usually the file must be recataloged rather than extended. One method is to use the COPY and COMBINE utilities as illustrated below.

```
PNC
Job Card
ATTACH,A,MYPF.
SKIPF,A,999,17.
COPYCR,INPUT,A.
REWIND,A.
COMBINE,A,B,999.
CATALOG,B,MYPF2,RP=999.
PURGE,A.
7/8/9
     data
6/7/8/9
```

The SKIPF card positions A to end-of-information (assuming there are not more than 999 EOF marks), and the COPYCR card copies the data cards to the end of A. Next the COMBINE card combines all logical records on file A (assuming there are not more than 999) and writes them as a single logical record on file B. The last cards purge file A and catalog file B in its place.

5.2.5 PNPURGE

PNPURGE permits the user to purge unwanted permanent files without having to attach the files or supply any passwords. PNPURGE only requires that the file "belong" to the user. More specifically, a problem number manager is allowed to purge any file cataloged under his/her PN, while a non-PN manager can only purge the files cataloged under his/her user ID.

PNPURGE is called as follows:

```
PNPURGE[,PFN=pfn[,DPFN=dpfnd][,CY=nn][,I=lfn].
```

PFN=pfn — specifies the permanent file name (pfn). The name may be as long as 40 characters and may include any characters other than a comma, period or parentheses.

DPFN=dpfnd specifies the permanent file name, where pfn is the permanent file name and d is any character the user selects to be a delimiter. Note that PNPURGE is the only PF control card which allows user defined delimiters. The delimiter character must appear immediately after the equal sign, and it may be any character (including the blank) that does not occur in the permanent file name. All characters between the first and second occurrence of the delimiter—including blanks, commas, periods and parentheses—are treated as part of the name (except in an interactive job, where the occurrence of a period or right parenthesis signals the end of a command.)

CY=nn specifies the cycle number (nn) of the file to be purged. Single digit cycle numbers may be specified with or without a leading zero. If this parameter is omitted, the highest numbered cycle of the file is assumed.

I=lfn specifies an input file (lfn) from which PNPURGE is to read the names and cycle numbers of the files to be purged. This option may be used separately or in combination with the PFN or DPFN parameter in order to specify several permanent file names. The format of the PNPURGE input file is compatible with unheadered output from PFLIST, see section 5.3. The input file should consist of card images having the following format:

cols. 3-42	permanent file name (left-justified)
cols. 55-56	cycle number (single-digit numbers may be right- or left-justified)

All parameters are optional. The control card 'PNPURGE.' is equivalent to 'PNPURGE,I=INPUT.'

The errors "PERMANENT FILE NOT ON SYSTEM" and "CYCLE REFERENCED DOES NOT EXIST" will not cause an abort. Other errors, such as trying to purge a file belonging to another user or trying to purge a file which is already attached, will cause an abort only after PNPURGE has attempted to complete all requested functions.

Example 1:

```
PNPURGE,DPFN=$FANCY PF. NAME$,CY=2.
```

This card purges cycle 2 of FANCY PF. NAME, provided that the file was cataloged by a job using the same ID (or for a master ID, by a job using the same PN).

Example 2:

```
PNC
USER,JC300,T30.
PFLIST,U=NAMES,SIZE=500.
REWIND,NAMES.
PNPURGE,I=NAMES.
6/7/8/9
```

This job uses PFLIST (see Section 5.3) to generate a list of the user's permanent files that are 500 PRUs or larger. This list, which is written to file NAMES, is properly formatted for input to PNPURGE.

5.3 PFLIST

Users may generate a report describing the status of their permanent files by executing PFLIST, an MSU utility which searches the permanent file tables for information about selected files.

Ordinarily, PFLIST lists just the files that "belong" to the user. That is, a PN manager will receive a list of all permanent files charged to the PN dollar balance, but a non-PN manager will receive a list of only the files charged to his/her ID dollar balance. Information for files cataloged with other PNs or PN ordinals may be requested by using one of the many PFLIST control card options.

The content and format of the PFLIST report is determined by whether the program is executed in batch or interactive mode, as well as by control card options. Detailed format specifications are provided in Tables 5-1 and 5-2, and a sample report from a PFLIST batch run is shown in Figure 5-1.

Because of the large number of control card options, the format of the PFLIST card is given as,

```
PFLIST[,p1,p2, ... ,pn].
```

If there are no parameters specified, PFLIST lists all active permanent files that belong to the user, as explained above. The parameters, p1 through pn, are described below in two categories: those that control the format of the PFLIST report, and those that specify selection criteria for the files listed. All parameters are optional and may be specified in any order.

Format Control

O=lfn specifies the file on which PFLIST is to write a labeled report (i.e., containing page headers). If O=lfn is omitted, the default file is OUTPUT for batch jobs and TTYTTY for interactive jobs (which means that output will not be listed at the terminal until PFLIST has ended). If the O parameter appears alone, O=ZZZZOT is assumed for interactive jobs (where ZZZZOT is connected). O=0 directs that labeled output not be generated.

U=lfn specifies the file on which PFLIST is to write an unlabeled report. This file will always be in BATCH format. This output is more suitable for input into another program. If the U parameter appears alone, U=UNHEAD is assumed.

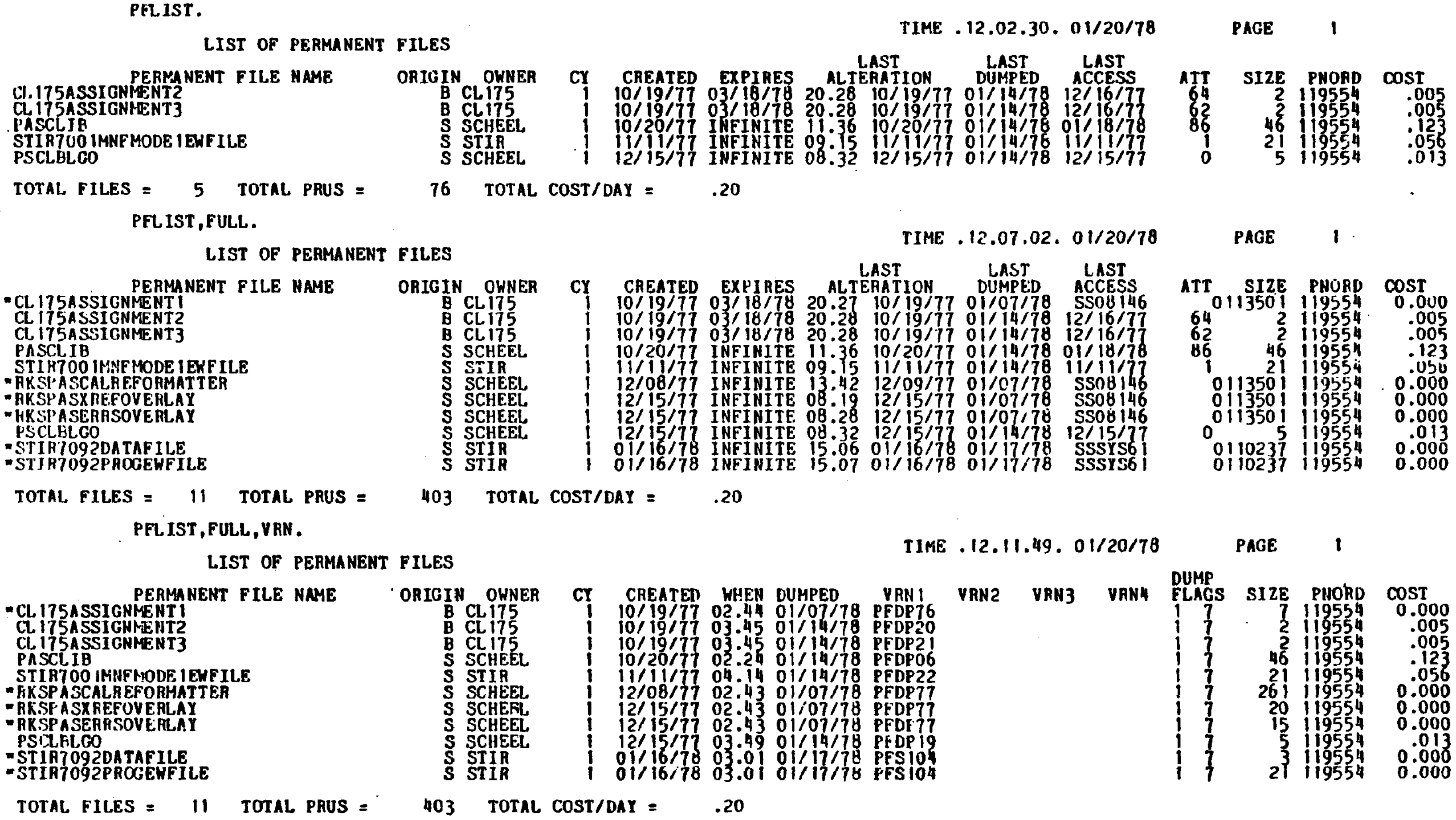
FDO=lfn designates the file on which PFLIST will write a full detail output file. All information (except dump information) about a particular PF will be listed on one very long (400 characters maximum) line. This file is intended for user programs to process and extract the desired information. This file is not normally generated. (See Table 5-3 for column specifications). FDO alone is illegal.

TTY write labeled output files in a format suitable for a 72-column terminal (see Table 5-2). TTY is assumed for interactive jobs, unless the BATCH parameter is specified.

BATCH write labeled output files in a format suitable for the line printers (see Table 5-1). BATCH is assumed for batch jobs unless the TTY parameter is specified. This format may also be used for terminals with a wide carriage (136 column).

VRN list the VRN(s) of the tape(s) on which each permanent file was last dumped. This information is printed in place of the last alteration, last access, and number of attaches data.

Figure 5—1



```
       PFLIST.
                                                                                 TIME .12.02.30. 01/20/78        PAGE      1
             LIST OF PERMANENT FILES
                                                                             LAST         LAST     LAST
       PERMANENT FILE NAME         ORIGIN OWNER     CY  CREATED  EXPIRES   ALTERATION     DUMPED   ACCESS    ATT    SIZE PNORD    COST
 CL175ASSIGNMENT2                       B CL175     1  10/19/77 03/18/78 20.28 10/19/77 01/14/78 12/16/77    64       2 119554   .005
 CL175ASSIGNMENT3                       B CL175     1  10/19/77 03/18/78 20.28 10/19/77 01/14/78 12/16/77    62       2 119554   .005
 PASCLIB                                S SCHEEL    1  10/20/77 INFINITE 11.36 10/20/77 01/14/78 01/18/78    86      46 119554   .123
 STIR7001MNFMODE1EWFILE                 S STIR      1  11/11/77 INFINITE 09.15 11/11/77 01/14/78 11/11/77     1      21 119554   .056
 PSCLBLGO                               S SCHEEL    1  12/15/77 INFINITE 08.32 12/15/77 01/14/78 12/15/77     0       5 119554   .013

 TOTAL FILES =     5   TOTAL PRUS =       76   TOTAL COST/DAY =      .20
       PFLIST,FULL.
                                                                                 TIME .12.07.02. 01/20/78        PAGE      1
             LIST OF PERMANENT FILES
                                                                             LAST         LAST     LAST
       PERMANENT FILE NAME         ORIGIN OWNER     CY  CREATED  EXPIRES   ALTERATION     DUMPED   ACCESS    ATT    SIZE PNORD    COST
*CL175ASSIGNMENT1                       B CL175     1  10/19/77 03/18/78 20.27 10/19/77 01/07/78  SS08146       0113501 119554  0.000
 CL175ASSIGNMENT2                       B CL175     1  10/19/77 03/18/78 20.28 10/19/77 01/14/78 12/16/77    64       2 119554   .005
 CL175ASSIGNMENT3                       B CL175     1  10/19/77 03/18/78 20.28 10/19/77 01/14/78 12/16/77    62       2 119554   .005
 PASCLIB                                S SCHEEL    1  10/20/77 INFINITE 11.36 10/20/77 01/14/78 01/18/78    86      46 119554   .123
 STIR7001MNFMODE1EWFILE                 S STIR      1  11/11/77 INFINITE 09.15 11/11/77 01/14/78 11/11/77     1      21 119554   .056
*RKSPASCALREFORMATTER                   S SCHEEL    1  12/08/77 INFINITE 13.42 12/09/77 01/07/78  SS08146       0113501 119554  0.000
*RKSPASXREFOVERLAY                      S SCHEEL    1  12/15/77 INFINITE 08.19 12/15/77 01/07/78  SS08146       0113501 119554  0.000
*RKSPASERRSOVERLAY                      S SCHEEL    1  12/15/77 INFINITE 08.28 12/15/77 01/07/78  SS08146       0113501 119554  0.000
 PSCLBLGO                               S SCHEEL    1  12/15/77 INFINITE 08.32 12/15/77 01/14/78 12/15/77     0       5 119554   .013
*STIR7092DATAFILE                       S STIR      1  01/16/78 INFINITE 15.06 01/16/78 01/17/78  SSSYS61       0110237 119554  0.000
*STIR7092PROGEWFILE                     S STIR      1  01/16/78 INFINITE 15.07 01/16/78 01/17/78  SSSYS61       0110237 119554  0.000

 TOTAL FILES =    11   TOTAL PRUS =      403   TOTAL COST/DAY =      .20
       PFLIST,FULL,VRN.
                                                                                 TIME .12.11.49. 01/20/78        PAGE      1
             LIST OF PERMANENT FILES
                                                                                                                 DUMP
       PERMANENT FILE NAME         ORIGIN OWNER     CY  CREATED  WHEN DUMPED     VRN1     VRN2     VRN3     VRN4  FLAGS    SIZE  PNORD   COST
*CL175ASSIGNMENT1                       B CL175     1  10/19/77 02.44 01/07/78 PFDP76                             1   7       7 119554  0.000
 CL175ASSIGNMENT2                       B CL175     1  10/19/77 03.45 01/14/78 PFDP20                             1   7       2 119554   .005
 CL175ASSIGNMENT3                       B CL175     1  10/19/77 03.45 01/14/78 PFDP21                             1   7       2 119554   .005
 PASCLIB                                S SCHEEL    1  10/20/77 02.24 01/14/78 PFDP06                             1   7      46 119554   .123
 STIR7001MNFMODE1EWFILE                 S STIR      1  11/11/77 04.14 01/14/78 PFDP22                             1   7      21 119554   .056
*RKSPASCALREFORMATTER                   S SCHEEL    1  12/08/77 02.43 01/07/78 PFDP77                             1   7     261 119554  0.000
*RKSPASXREFOVERLAY                      S SCHERL    1  12/15/77 02.43 01/07/78 PFDP77                             1   7      20 119554  0.000
*RKSPASERRSOVERLAY                      S SCHEEL    1  12/15/77 02.43 01/07/78 PFDP77                             1   7      15 119554  0.000
 PSCLBLGO                               S SCHEEL    1  12/15/77 03.49 01/14/78 PFDP19                             1   7       5 119554   .013
*STIR7092DATAFILE                       S STIR      1  01/16/78 03.01 01/17/78 PFS104                             1   7       3 119554  0.000
*STIR7092PROGEWFILE                     S STIR      1  01/16/78 03.01 01/17/78 PFS104                             1   7      21 119554  0.000

 TOTAL FILES =    11   TOTAL PRUS =      403   TOTAL COST/DAY =      .20
```

Table 5—1

PFLIST `BATCH' Formats

Normal Format

column	contents
1	carriage control
3-42	permanent file name
43	origin (source code character)
45-53	owner ID (from CATALOG card)
55-56	cycle number
57	cycle altered flag (* if altered)
59-66	creation date (or date of last catalog if `LASTCAT' specified)
68-75	expiration date
77-81	time of last alteration (or time of last catalog i `LASTCAT' specified)
83-90	date of last alteration
92-99	date file was last dumped to tape
101-108	date of last access
111-114	number of attaches
116-121	size of file in PRUs
123-128	PN ordinal (identifies user ID)
130-136	cost per day (or PN if `LISTPN' is specified)

Entries for purged files (`FULL' or `PURGE' specified) have th following differences.

2	purged file flag (* if purged)
102-108	sequence number of the job that purged the file
115-121	problem number of the job that purged the file

VRN Format (`VRN' or `FULLVRN' selected)

column	contents
68-72	time of dump
74-81	date of dump
83-88	VRN1
90-95	VRN2 (first overflow reel)
97-102	VRN3(second overflow reel)
104-109	VRN4 (third overflow reel)

110-111	number of VRNs used for dump
112	verification status
113	dump obsolete flag
114	7 or 9-track tape flag (7/9)
117-121	size of file in PRUs

If there are more than 4 VRNs used for the dump, additional lines are printed listing the extra VRNs only.

83-88	VRN (I)
90-95	VRN (I+1)
97-102	. .
104-109	. .

Table 5—2

PFLIST `TTY' Format

Normal Format

column	contents
1	carriage control
3-42	permanent file name. If the permanent file name is longer than 20 characters, the remaining information is printed on a separate line.
24-25	cycle number
26	cycle altered flag (* if altered)
27-35	owner ID (from CATALOG card)
37-41	creation date (or date of last catalog if `LASTCAT' is specified)
43-47	expiration date
49-53	date of last access
54-57	number of attaches
58-63	size of file in PRUs
65-71	cost per day (or PN if `LISTPN' is specified)

Entries for purged files (`FULL' or `PURGE' specified) have the following differences.

2	purged file flag (* if purged)
49-53	date of last alteration
56-62	sequence number of the job that purged the file
64-70	problem number of the job that purged the file

VRN Format (`VRN' or `FULLVRN' selected)

column	contents
27-31	date of dump
33-38	VRN1
40-45	VRN2 (first overflow reel)
47-52	VRN3 (second overflow reel)
54-59	VRN4 (third overflow reel)
60-61	number of VRNs used for dump
62	verification status
63	dump obsolete flag
64	7 or 9-track tape flag (7/9)
67-71	size of file in PRUs

If there are more than 4 VRNs used for the dump, additional lines are printed listing the extra VRNs only.

Table 5—3

Full Detail Output (FDO) Format

Primary Entry for Permanent File (Line 1)

column	contents
1-136	identical to standard batch format with `LISTPN' selected, `LASTCAT' not selected, and columns 102-121 not modified for purged files.
151-153	number of dump history entries (i.e. number of continuation lines for this PF)
155-157	random file flag (`RND' if random)
160-162	the characters CR=
163-170	creation time
172-174	the characters BC=
175-176	BC count from the CATALOG card
178-181	the characters CAT=
183-187	time of last catalog
189-196	date of last catalog
198	the character $
199-205	cost per day
207-210	the characters PRG= (for purged files only)
211-217	sequence number of the job that purged the file
219-225	problem number of the job that purged the file

Dump History Output
One line is generated for each dump history entry.

2	the character +
11-16	the characters DUMPED
18-22	time of dump
24-31	date of dump
33-35	position of dump on first VRN
37	verification status
38	dump obsolete flag
39	7 or 9-track tape flag (7/9)
41-42	number of VRNs used for dump
44-133	VRNs of the first 13 dump tapes (listed in 6 character columns, separated by blanks)
156-273	VRNs 14-30
296-399	VRNs 31-45

FULLVRN list full dump history for each file. A line giving the dump date/time and the dump tape VRN is printed for each time the file was dumped.

SORT=key sort the output (on the file specified by O=lfn or U=lfn) as indicated by key, where key is one of the following keywords.

ALPHA	sort by permanent file name
ACCOUNT	sort by PN and PN ordinal within the PN
PN	sort by PN
SIZE	sort by size (smallest first)
LASTACC	sort by last access (most recent first)
0	no sort

SORT=ALPHA is assumed if the SORT keyword appears alone. SORT=0 is assumed if the SORT option is omitted.

LISTPN print the PN of the PF owner for each file in place of the cost.

LASTCAT print the date on which the file was last cataloged in place of the creation date and the time when last cataloged in place of the last alteration time.

Note: The 'VRN', 'FULLVRN' and 'LASTCAT' parameters are mutually exclusive.

File Selection

PFLIST determines which files are to be listed by performing a logical AND on the specified search criteria. For example, PFLIST(PREFIX=ABC,SIZE=100) says: "List all permanent files whose names begin with 'ABC' and that are 100 PRUs or more and that belong to this user."

MT=vrn=...
Search the specified 7-track PF dump tape set rather than the system RBTC tables.

NT=vrn=...
Search the specified 9-track PF dump tape set rather than the system RBTC tables.

Any number of PF dump tape sets may be specified using multiple occurrences of the MT and NT parameters.

PFN=pfn
List only the specified PF. This parameter will accept '$' delimiters when special characters are included in the permanent file name. A '$' within the name must be represented as '$$'.

PREFIX=prefix
List the permanent files whose names begin with the characters specified. A prefix of up to 40 characters may be specified. This parameter will accept '$' delimiters when special characters are to be in the prefix. A '$' within the prefix must be represented as '$$'.

ID=id1[=id2=id3=id4=id5]
List only the files cataloged with one of the IDs specified. This does not refer to the ID of the job card, but to the ID=name parameter of the CATALOG card.

SOURCE[=ABC ... Z12 ... 0]
List only the files that were cataloged from the source specified (see Appendix E). If the keyword SOURCE is used alone, SOURCE=B is assumed.

SIZE=pru
List only the files that are larger than or equal to the number of PRUs specified.

ATTACH[=attaches]
List only the files whose attach count is less than or equal to that specified. If the ATTACH keyword appears alone, ATTACH=0 is assumed.

DUMPED[=mm/dd/yy]
List those files dumped on or after the specified date. If no date is specified, the current date is assumed.

ALTERED[=mm/dd/yy]
List only the files that have been altered on or after the date specified. If no date is specified, list only the files that have been altered since their last dump date.

LASTALT[=mm/dd/yy]
List only the files that were last altered on or before the date specified. If no date is specified, the current date is assumed.

ACCESS[=mm/dd/yy]
List only the files that have been attached on or after the date specified. If no date is specified, the current date is assumed.

LASTACC[=mm/dd/yy]
List only the files that were last accessed on or before the date specified. If no date is specified, the current date is assumed.

EXPIRED[=mm/dd/yy]
List only the files that expire on or before the date specified. If no date is specified, the current date is assumed.

PURGED
List only the files that have been purged (since the last initial deadstart or within approximately three weeks; see Section 1. 7).[4]

FULL
List both purged and active files.[4]

ALL
List all files satisfying the options specified, including those cataloged under different PNs. If ALL is the only parameter specified, all permanent files in the system will be listed.

PN[=pn1= ... =pn5]
List only these files that have been cataloged under the PNs specified. If the PN keyword appears alone, the PN of the user is assumed. "PN" is the default option for problem number managers unless the ALL, PNDEPT, or an explicit PN or PNORD parameter is used.

PNORD[=pnord1= ... =pnord5]
List only those files that have been cataloged under the PN ordinals specified. The PN ordinal uniquely identifies each PN subaccount (user ID) in the Authorization File. If the PNORD keyword appears alone, the PN ordinal of the user is assumed. This is the default option for non-PN managers unless the ALL, PNDEPT, PN, or an explicit PNORD parameter is used. The PNORD can be obtained with the AUTHORF utility.

[4] Information for purged files is lost after an initial deadstart.

PNDEPT[=dept]

List only the files cataloged under department code dept, where dept is the first two characters of the problem number. If the PNDEPT keyword appears alone, the user's department code is assumed.

BACKUP [=U]

If the BACKUP keyword appears alone, list only the files which are not adequately backed up. Only verified dumps which are not obsolete are counted. If BACKUP=U is specified, unverified dumps will also count.

FULLDMP

If FULLDMP is specified, list all files for which BC≠0 (i.e. all files which are backed up by the Computer Laboratory) is specified.

Default Parameters

From batch, the control card 'PFLIST.' is equivalent to

```
PFLIST,O=OUTPUT,SORT=0,BATCH,PNORD.
```

In interactive use, the command 'PFLIST.' is equivalent to

```
PFLIST,O=TTYTTY,SORT=0,TTY,PNORD.
```

Exception: PNORD is replaced by PN for jobs run with a master ID.

Examples:

1.
```
PFLIST.
```

Lists all files belonging to the user. If entered from a terminal, the report will be printed at the terminal in an abbreviated format.

2.
```
PFLIST,PREFIX=$J.A.L.$,ALL.
```

Lists all files whose names start with the characters 'J.A.L.'

3.
```
PFLIST,ALL,FULL,LISTPN.
```

Lists all permanent files found in the tables, both active and purged. The PN number for each file will be printed in place of the cost/day. WARNING: this job is very expensive.

4.
```
PFLIST,PNDEPT=01,SORT=SIZE.
```

Lists all permanent files cataloged by PNs assigned department code 01 (Computer Laboratory). The files will be sorted by size from smallest to largest.

5.
```
PFLIST,EXPIRED,PN=018072=018073=018074.
```

Lists all files cataloged under problem number 018072, 018073, or 018074 and that expire today (or that have already expired but for some reason have not been purged).

6.
```
PFLIST,U=FILE,O=0,SIZE=100,LASTACC=03/01/77,ATTACH=0.
```

Lists all files belonging to the user which are 100 PRUs or larger, which were last used on or before March 1, 1977, and which were never attached. The report is written on file FILE in an unlabeled format, which would be suitable for input to the PNPURGE program.

7. PFLIST,PURGED,VRN.

   Lists all files belonging to the user that have been purged. In general, information for purged files is retained for only as long as the files are backed up on permanent file dump tapes. Normally the report would give the sequence number of the job which purged the file, the PN of the job, and the date and time the file was purged. But because the VRN option is selected, dump tape information will be printed instead.

8. PFLIST,NT=UP2021=UP2022.

   Lists all files belonging to the user which are on the 9-track PF dump tape set UP2021 and UP2022.

5.4 PF Backup

Occasionally, permanent files can be lost due to hardware failure, software problems, etc. Permanent files are protected against most forms of accidental destruction, but they will be lost in any system failure which involves an **initial deadstart** (see Section 1.4.7) to reload the system. An initial deadstart erases all tables, including the Permanent File Directory and the RBTC. For this reason the Computer Laboratory tries to keep a current, magnetic tape copy of every permanent file on a temporary basis. The user can maintain privately owned backup tapes, rely on Computer Laboratory backup tapes or have no backup tapes at all. It is recommended that users maintain their own backup copies of infrequently used files, using the PFDUMP utility. The user can opt for no Computer Laboratory backup tapes by specifying BC=0 on the CATALOG card.

At the end of the daily production schedule, every permanent file that has been created or changed during that day is normally **dumped** (copied) to tape. The Computer Laboratory backup copy of a PF will be up to one day old. In case of accidental destruction the user can reload the PF from the Computer Laboratory tape within a two week period. After two weeks, it is still possible that a copy of the file exists. It may, however, be an older version. To locate such a copy, the user should contact the Operation Shift Supervisor. Periodically all permanent files are dumped and a new set of daily dumps is begun.

Another set of tapes is used to hold backup copies of permanent files purged by the Computer Laboratory. At the end of the production day, each permanent file is examined to determine if (1) the retention period has expired, (2) the owner's PN has expired, or (3) the owner's PN or ID dollar balance has been expended. Files that fall into any of these categories are dumped to tape and then purged.

The following sections described the Computer Laboratory policies and procedures for dumping, reloading, and purging permanent files. In addition, Section 5.4.3 explains how to use the PFLOAD control card to reload permanent files that have been purged or lost due to either system failure or user error. Section 5.4.4 describes how the user may use the PFDUMP utility to dump PFs from disk and/or existing dump tapes onto new dump tapes. The PFDUMP utility allows the user to maintain his/her own library of backup tapes.

5.4.1 Computer Laboratory Dumping/Loading Policies

All permanent files (which are not cataloged with a BC=0 parameter) are copied to a backup tape on a regular basis. Files that have been changed or created during a day's production are copied to a backup tape daily at the end of the production schedule. PFs are also dumped at the end of each quarter. Copies of Quarter Dumps are kept for two terms. Files may also be dumped at other times for special reasons, such as disk maintenance.

NOTE: A permanent file cannot be dumped if it is attached to a user's job with more than read permission. Users will be notified prior to any dump operation so that they can return files they wish to have dumped.

In the event of a system failure or user error which results in the loss of permanent files, only the files created during the current production day should be lost. All other files will be recreated from the backup tapes.

5.4.2 Purging Policies

Permanent files are dumped to tape and then purged in any of the following cases.

1. Expired permanent files are dumped and purged **on the day of their expiration** at the end of the production schedule on weekdays. EXCEPTION: Expired APLIB files are purged only if they have not been attached during the past four days. The default retention period for APLIB files is one day. Remember that the default retention period for other permanent files is 15 days. By specifying RP=0 on the CATALOG card, the user can have the file purged at the end of the day on which it is cataloged.

2. Unexpired permanent files are dumped and purged if

    a. the user (ID) dollar balance is less than or equal to zero and the file is not owned by a Master ID; or

    b. the PN dollar balance is less than or equal to zero and the file is not owned by a Master ID; or

    c. the file is owned by the Master ID and the PN dollar balance is less than or equal to -$10.

Note: Permanent files which belong to a user ID which has been deleted by the PN manager will not be automatically purged.

3. If a PN is expired, all files charged to the PN are dumped and purged at the end of the production day following the PN expiration date.

    The Computer Laboratory will not provide any backup service for files cataloged with BC=0.

Files that have been dumped and purged for any of these reasons are kept for a minimum of 14 days. The user can locate the VRN of the purged files using PFLIST and can recreate the file by using the PFLOAD control card, as described below.

5.4.3 PFDUMP

PFDUMP allows the user to create a new dump tape or tape set by dumping selected PFs from disk and/or existing PF dump tapes onto the new dump tape set. Existing tape sets are called the "old" tape sets. The tape set which is being created is called the "new" tape set. PF dump tapes all have a special VRN of the form UPnnnn. The user can purchase special PF dump tapes at the Service Window in the I/O room. The user can also transfer regular tapes to UPnnnn tapes. It normally takes 24 hours to process a tape transfer.

For security reasons, PF dump tapes can be accessed only by the PF utilities (PFDUMP, PFLOAD and PFLIST). **These tapes will not be released to the owner until they have been completely erased by the Computer Laboratory, since they may contain sensitive information and PFs which belong to other people.**

PFDUMP allows users to maintain their own PF backup, independent of the normal PF backup procedure followed by the Computer Laboratory. PF backup tapes can be kept for as long as the user wishes. Large infrequently used files can be kept on tape rather than disk. This reduces the cost to the user and frees valuable disk space. The user can update and edit his/her own backup tapes.

The PFDUMP control card is of the following form:

```
PFDUMP [,p1,p2, ... ,pn].
```

The control card parameters are divided into five groups:

Tape Specification
Disk Specification
Output File Specification
Specification of Existing PF Dump Tapes
Specification of PFs to Copy from Existing PF Dump Tapes

Tape specification:

MT=vrn=... specifies which tapes are to be written; this applies only to 7-track tapes. MT alone is illegal.

NT=vrn=... specifies which tapes are to be written; this applies only to 9-track tapes. NT alone is illegal.

NEWPN=nnnnnn specifies the PN to be written on the new tape labels. 'nnnnnn' may be a 6 or 7 digit problem number. If NEWPN is not specified, the PN under which the job is running is used. If NEWPN=0, any subsequent job will be allowed to rewrite the new tapes. Any non-zero value for NEWPN restricts the ability to rewrite the new tapes to the designated PN.

Note: Up to 62 VRNs may be specified. The 'MT' and 'NT' parameters are mutually exclusive, but one of them must be supplied. No VRN may appear more than once (i.e. new dump tapes and old dump tapes must be mutually exclusive).

Disk specification:

PFN=xx specifies a single PF to be dumped. The PFN may be delimited '$', allowing the use of special characters within the PFN (in which case a '$' within the PFN must be represented by '$$').

CY=xx allows a cycle number to be specified in conjunction with the PFN parameter, and is illegal unless the PFN parameter is also used. Legal forms are:

CY=xx where xx is the cycle number to be dumped 1≤xx≤63.

CY=ANY causes the first cycle encountered on disk to be dumped. If CY is omitted, CY=ANY is assumed.

CY=ALL causes all cycles of that PFN to be dumped.

ADD=lfn=... specifies up to 5 lfns which hold a list of the PFs to be dumped from disk onto tape. If ADD appears alone, ADD=INPUT is assumed. (See PFDUMP Input Lists, for format specifications.)

WAIT specifies that PFDUMP should wait for any PFs that are in use by another job. Use of the WAIT parameter does not ensure that the PFs will be dumped in the order specified in the ADD list. If 'WAIT' is not specified, PFDUMP will skip the file in use, noting the action in the output file.

ORDER specifies that the disk PFs are to be dumped in the order in which they are specified in the ADD list. This is normally the case, but if PFDUMP must wait for a PF that is in use, the order might not be retained without the use of this parameter.

Output file specification:

O=lfn specifies the file upon which PFDUMP will echo all input lists (along with appropriate diagnostics) followed by a commentary on every PF that is processed. If O is not specified or O appears alone, O=OUTPUT is assumed.

U=lfn specifies the file on which PFDUMP will write an unheadered output file containing one line for each PF that it processes. The format and content is the same as the 'O' file without the echo of input lists. If U is not specified, no such file is generated. If U appears alone, U=UNHEAD is assumed.

IX=lfn specifies the file on which PFDUMP will write an index output file, which is a list of those PFs which were actually dumped on the new tapes. PFs are listed in the order that they occur on the new dump tapes, along with information about the tapes on which they reside.

Note: The output files use the following PFLIST compatible format.

Col. 3-42	PFN.
Col. 54-56	cycle number.
Col. 58-134	commentary; code number, description of what was done with the PF, a list of VRNs on which this PF was dumped.

Specification of existing PF dump tapes:

OLDMT=vrn=... specifies the VRNs of one or more existing 7-track PF dump tapes that are to be merged with the PF's dumped from disk. OLDMT alone is illegal.

OLDNT=vrn=... specifies the VRNs of one or more existing 9-track PF dump tapes that are to be merged with the PF's dumped from disk. OLDNT alone is illegal.

Note: 'OLDMT' and 'OLDNT' may be specified up to twenty times, with each occurrence specifying the VRNs of exactly one set of PF dump tapes that were written at the same time as continuation reels. Up to 62 VRNs may be specified with each occurrence of the 'OLDMT' or 'OLDNT' parameters. The VRNs must be specified in the same order as when they were written. If neither are specified, only PFs from disk will be written on the new dump tapes.

Specification of PFs to copy from existing PF dump tapes:

DROP=lfn=... specifies up to 5 lfns which hold a list of the PFs to be skipped during the copying of the old dump tapes to the new dump tapes. All PFs not on this list will be copied onto the new dump tapes unless they already exist on the new dump tapes. (See PFDUMP Input Lists for format specification.)

KEEP=lfn= ... specifies up to 5 lfns which hold a list of PFs to be copied from the old dump tapes to the new dump tapes.PFs not on this list will not be copied. (See PFDUMP Input Lists for format specification)

Note: 'DROP' and 'KEEP' are mutually exclusive. Neither parameter has a default value. If the user does not supply a DROP or KEEP list, all PFs on the tapes are copied.

PFDUMP Input Lists:

The input lists (for the 'ADD', 'KEEP', and 'DROP' parameters) may be in either of two formats.

PFDUMP will detect the proper format when it reads the input files.

One format is called **PFLIST format**, since it is compatible with unheadered PFLIST output. The relevant fields in this format are:

Col.	1	must be blank to indicate PFLIST format.	
Col.	3-42	permanent file name.	
Col.	54-56	cycle specification (right justified): Legal entries are:	
		xx	dump cycle number xx where $1 \leq xx \leq 63$.
		ALL	dump all cycles of the PF
		ANY	copy the first cycle encountered on tape; or dump the highest cycle on disk.
		blank	same as ANY

The other format is **keyword format**, since it uses a control card type of keyword format. Each card must start in column 1 (to distinguish this format from PFLIST format) and looks like:

pfn[,CY=xx]

or if the permanent file name contains special characters,

pfn[,CY=xx]

The PFN or the '$' delimiter **must** start in column 1. A '$' within a delimited PFN must be represented as '$$'. CY=xx is an optional parameter which specifies the cycle to be dumped. The legal forms of this parameter are the same as for PFLIST format, above.

The format used by one input list does not affect the formats of the other input lists (if any); however, all cards on the same PFDUMP input list should follow the same format. PFDUMP will accept mixed formats but will issue a warning message that the format was not consistent. This is intended to highlight possible errors in the preparation of the input lists.

PFDUMP Output Files:

There are three types of PFDUMP output files: the **general output file** specified by the O parameter, the **unheadered output file** specified by the U parameter, and the **PF index output file** specified by the IX parameter.

The **general output file** contains the following information:

1. an echo of the PFDUMP control cards.

2. an echo of all input cards along with any error diagnostics which may have been generated.

3. PFDUMP result list, which contains information on all PFs processed by PFDUMP. This information is identical to the information produced for an unheadered output file.

4. a short tape usage estimate; giving the number of PRUs written on the last reel, the estimated number of feet of tape used, the estimated number of PRUs of PFs which would fill the remaining space.

The **unheadered output file** is in machine readable format and contains one or more lines of output for each PF processed by PFDUMP. The format is as follows:

Entries for all files

column	contents
3-42	permanent file name.
45-53	owner ID.
54-56	cycle number, ANY or ALL.
58-62	time of last alteration.
64-71	date of last alteration.
130-131	A code number which indicates the action taken with this PF. See Table 5-4.

Entries for files which were not dumped or copied

73-85	the characters 'NOT DUMPED-', if the PF were to be dumped from disk; or the characters 'NOT COPIED-', if the PF were to be copied from tape.
86-122	one of the following explanatory messages: NO ROOM ON NEW TAPES PF ALREADY ON TAPE PFN,CY ON DROP LIST PFN,ANY ON DROP LIST PFN,ALL ON DROP LIST DEFAULT DROP (This PFN and cycle did not appear in KEEP input list.) ERROR IN PF HEADER - PF UNKNOWN INCOMPLETE CYCLE FILE NOT IN SYSTEM CYCLE NOT IN SYSTEM ALREADY ATTACHED BY THIS JOB (This PF is attached by this job and cannot be dumped.) IN USE BY OTHER JOB (This PF is in use by another job with more than read-only permission, and PFDUMP has not been told to wait for it.) ATTACH ERROR CODE = NNN

Entries for files which were dumped or copied have the following differences:

73-81	the character string DUMPED TO or COPIED TO depending on whether the PF was dumped from disk or copied from tape.
83-88	first VRN to which the PF was dumped/copied.
90-95	continuation VRN (if any).
97-102	continuation VRN (if any).
104-109	continuation VRN (if any).
110-111	number of VRNs used to dump/copy this PF.
114	7 or 9 track tape flag.
115-120	size of the PF in disk PRUs.
122-128	possible warning message: blank no errors *ERROR* disk or tape parity error.

Table 5—4

CODE	MEANING
1	PF was dumped from disk without problem.
2	PF was dumped from disk with one or more disk parity errors.
3	PF was copied from tape without problem.
4	PF was copied from tape with one or more errors detected on the old dump tape.
5	PF was not copied from the old dump tape because that PFN and cycle were specified in the DROP input list.
6	PF was not copied from the old dump tape because that PFN and a cycle of ANY were specified in the DROP input list, and this is the first cycle of that PFN that has been encountered during the tape-to-tape copy.
7	PF was not copied from the old dump tape because that PFN and a cycle of ALL were specified in the DROP input list.
8	PF was not copied from the old dump tape by default (i.e., the PF was not specified in the KEEP input list.)
9	PF was not dumped from disk or was not copied from the old dump tape because that PFN and cycle has already been written on the new dump tape set.
10	PF was not dumped from disk or was not copied from the old dump tape because that PFN and cycle has already been written on the new dump tape set.
11	PF was not dumped from disk because no such PFN exists on the system.
12	PF was not dumped from disk because the specified cycle of that PFN does not exist on the system.
13	PF was not dumped from disk because it was in use by another job and the WAIT parameter was not specified on the control card.
14	PF was not dumped from disk because it was already attached by this job.
15	PF was not dumped from disk because it is an incomplete cycle.
16	PF was specified in the KEEP input list but was never encountered on any old dump tape.
17	PF was specified in the DROP input list but was never encountered on any old dump tape.
18	An error was encountered on the old dump tape within a `PF header' As such, the PFN and cycle are unknown and that PF has been skipped. The PFN field of this line will hold something along the order of ** UNKNOWN PF **.
19	PF was not dumped from disk to tape because an unrecognized error occurred during the attach. The return code from the attach is also printed on this line.

Additional lines for files which were dumped or copied.

83-88	continuation VRN (if any)
90-95	continuation VRN (if any)
97-102	continuation VRN (if any)
104-109	continuation VRN (if any)

Entries for files which were on the DROP or KEEP input list that were not found on any old tape set have the following differences:

73-79	the characters WAS ON.
80-83	the characters DROP or KEEP.
84-134	the characters LIST BUT WAS NOT FOUND ON ANY OLD TAPE.

The **PF index output file** contains only entries for those PFs which were actually dumped or copied to the new PF tapes in the same format as the unheadered output file. This provides an index to the contents of the new tape set.

Note:

1. PFDUMP maintains no particular order on the new dump tapes. All disk PFs are first dumped to the new dump tapes, then the PFs from old dump tapes are copied to the new dump tapes. These copied PFs are taken in the order they are found on the tapes.

2. Each PFN and cycle number specifies a unique PF on the new dump tapes. PFDUMP will keep track of which PFs have already been written on the new dump tapes, and if that PFN and cycle number occur again it is either ignored (if it was to be dumped from disk) or automatically dropped (if it was to be copied from the old dump tapes). Thus a PF that was successfully dumped from disk will replace any copy of itself that might have been on the old dump tapes. However, if for any reason a PF should be unable to be dumped from disk, it will automatically be put on the 'KEEP' list, overriding any command to the contrary. This will ensure that a PF on the old dump tapes is not lost due to the disk copy being in use, etc.

Example 1:

The following job will make a backup tape copy of all permanent files belonging to the user's PN. The PFLIST card generates the list of permanent files. The PFDUMP card causes the files specified in LIST to be dumped to the PFDUMP tape UP1200. This job will wait for PFs which may be in use by another job.

```
PNC
id,MT1,...
PFLIST,U=LIST.
PFDUMP,MT=UP1200,ADD=LIST,WAIT.
6/7/8/9
```

Example 2:

In this example, disk files are merged with selected files from the old dump tape (UP1200) to produce a new dump tape (UP1201). The input list (in keyword format) specifies that PFDUMP copy all cycles of SAMPLEPF1, the first cycle of SAMPLEPF2 encountered on the tape, cycle two of SAMPLEPF3 and the first cycle encountered of MONEY$PF. This job prints a list of the contents of UP1201, sorted in alphabetical order.

```
PNC
id,MT2,...
PFLIST,U=LIST.
PFDUMP,ADD=LIST,MT=UP1201,WAIT,OLDMT=UP1200,KEEP=INPUT.
PFLIST,SORT,MT=UP1201.
7/8/9
SAMPLEPF1,CY=ALL
SAMPLEPF2
SAMPLEPF3,CY=2
$MONEY$$PF$
6/7/8/9
```

5.4.4 PFLOAD

The PFLOAD card reloads one or more permanent files from a dump tape. The file is reloaded with the cataloging information still intact.

The user has the option of dumping his/her own files onto tape. PFLOAD allows the user to reload information from both system and user dump tapes.

The PFLOAD control card looks like

```
PFLOAD [,p1,p2, ... ,pn].
```

The parameters are categorized into 4 groups.

Tape specification:

MT=vrn=... specifies a global tape set to be loaded; this applies only to 7-track tapes.

NT=vrn=... specifies a global tape set to be loaded; this applies only to 9-track tapes.

MT alone, indicates that the tapes whose VRN's are supplied in the input list are 7-track. (See PFLOAD Input files below.)

NT alone, indicates that the tapes whose VRN's are supplied in the input list are 9-track. (See PFLOAD Input files below.)

Note: Up to 62 VRN's may be specified in each global tape set. The global tape sets are used when reloading the PF specified by the PFN parameter or PFs designated by the input list with no local tape set specified. More than one global tape set may be specified, simply by specifying multiple occurrences of the MT and/or NT parameters.

Both 7-track and 9-track tape sets may be used on the same run.

PF selection:

ALL *reload all PFs on the specified tapes.*

I=lfn=... specifies up to 5 lfns which hold a list of the PFs to be reloaded. (See PFLOAD Input files below)

PFN=xx specifies a single PF to be reloaded. The pfn may be delimited by '$', allowing the use of special characters with the PFN (in which case a '$' within the PFN must be represented by '$$').

CY=xx allows a cycle number to be specified in conjunction with the PFN parameter, and is illegal unless the PFN parameter is also used. Legal forms are:

CY=xx where xx is the cycle number to be reloaded. 1≤xx≤63.

CY=ANY causes the first cycle encountered on the tape set to be reloaded. If CY is omitted, CY=ANY is assumed.

CY=ALL causes all cycles of that PFN to be reloaded.

Note: The 'ALL', 'I' and 'PFN' parameters are mutually exclusive.

Recataloging information.

RP=xx specifies the retention period of the reloaded PF's in days. Legal forms are:

RP=xx where 0 ≤xx≤999. If RP is omitted, RP=15 is assumed.

RP=SAME uses the RP value assigned to each PF before it was dumped.

DUP=xx specifies the action to be taken when PFLOAD attempts to recatalog a PF and finds that the PFN, or PFN and cycle number, are already in use by another disk PF. Legal forms are:

DUP=IGNORE causes PFLOAD to skip this PF (noting this action in the output file) and continue on to the next PF.

DUP=NEWNAME causes PFLOAD to create a new pfn, cataloging the PF under the new name. DUP=NEWNAME is assumed if DUP is not specified.

Output file specification:

O=lfn specifies the file upon which PFLOAD will echo the control cards, all input cards, and then list all the results of the reload attempts. If O is not specified, O=OUTPUT is assumed.

U=lfn specifies the file upon which PFLOAD will write an unheadered list of the results of the reload attempts. U alone implies U=UNHEAD.

Note: If U is omitted, no unheadered output is generated. Both headered and unheadered output lists use the following PFLIST compatible format.

Col.	3-42	pfn of PF as on tape.
Col.	48-52	The word CYCLE.
Col.	54-56	cycle of PF as on tape.

Col.	58-134	results: if the PF was not reloaded, this field indicates why. If the PF was reloaded, this field indicates any new pfn and cycle if one was created.

PFLOAD Input files:

All input files are rewound and read from the beginning-of-information to end-of-file, except for INPUT. The file INPUT will be read from the current position up to an end-of-record.

The input files may be in either of two formats. PFLOAD will detect the proper format when it reads the input files.

One format is called 'PFLIST format' since it is compatible with unheadered 'PFLIST,VRN' output. The relevant fields in this format are:

Col.	1	blank to indicate PFLIST format.	
Col.	3-42	permanent file name.	
Col.	54-56	cycle specification (right-justified). Legal options are:	
		xx	reloads cycle xx
		ALL	reloads all cycles of the PF
		ANY	reloads the first cycle encountered on the tape
		blank	same as ANY
Col.	83-88	VRN from which to reload this PF.	
Col.	90-95	continuation reel VRN (if any).	
Col.	97-102	continuation reel VRN (if any).	
Col.	104-109	continuation reel VRN (if any).	
Col.	114	tape type specification	
		7	specified 7-track
		9	specified 9-track
		blank	use the type specified on the control card. If no type was specified on the control card, '7' is assumed. If both MT and NT were specified an error diagnostic is issued and the input card is ignored.

The VRN field may contain one or more VRNs, in which case this PF may be reloaded only from the specified tape(s) (local tape set), or it may be blank, in which case this PF may be reloaded only from the tapes specified by the MT and/or NT parameters (global tape set). This is to allow for cases in which the PF may exist on several tapes that were dumped at different times, and a particular copy is to be retrieved. If any VRN field is blank, the rest of the continuation reel VRN's on this card are ignored.

The other format is 'keyword format', since it uses a control card type of keyword format. Each card must start in column 1 (to distinguish this format from PFLIST format) and looks like:

```
pfn,CY=xx,keyword=xx,keyword=xx...
```

or, if the permanent file name contains special characters,

```
$pfn$,CY=xx,keyword=xx,...
```

The permanent file name or the '$' delimiter must start in column 1. A '$' within a delimited PFN must be represented as '$$'.

CY=xx specifies the cycle or cycles to be reloaded. Legal forms are:

CY=xx where xx is the cycle number to be reloaded. 1≤xx≤63.

CY=ALL reloads all cycles of the PF

CY=ANY reloads the first cycle encountered on the tape. If CY is not specified CY=ANY is assumed.

MT=vrn=... the PF is reloaded from the specified 7-track tapes. MT and NT are mutually exclusive.

NT=vrn=... the PF is reloaded from the specified 9-track tapes. MT and NT are mutually exclusive

RP=xx specifies the retention period for this PF only. This overrides the RP parameter on the control card. If RP is omitted in the input file, the RP value on the control card is used. Legal forms are the same as the control card parameter.

DUP=xx specifies the action to be taken when PFLOAD attempts to recatalog this PF and finds that the pfn, or pfn and cycle number, are already in use by another disk PF. This overrides the DUP parameter on the control card. Legal forms are the same as the control card parameter. If DUP is omitted in the input file, the DUP value on the control card is used.

If the last non-blank character on a card is a comma, the next card will be read as a continuation card.

The format used by one input file does not affect the formats of the other input files (if any). However, all cards within any one PFLOAD input file should follow the same format. PFLOAD will accept mixed formats but will issue a warning message that the format was not consistent. This is intended to highlight possible errors in the preparation of the input lists.

Passwords:

PFLOAD will catalog a new cycle of an existing disk PF only if all passwords of the existing disk PF and the new PF are identical. This varies from the requirement of new cycle generation when a user catalogs a PF, in which the new cycle must get only Turnkey and Control permissions. Otherwise a PF reloaded and cataloged as a new cycle of an existing PF could possibly have its Read, Extend, and Modify passwords changed to those of the existing PF. This can cause problems:

1. A possible breach of security, since the owner of the existing PF can get all permissions to the new cycle.

2. The owner of the new cycle may not be able to access his/her PF due to the passwords having been changed.

Example 1:

In this example all files whose names begin with "COMMON PREFIX" on the 9-track PF dump tapes UP1000 or UP1001 are reloaded as permanent files with a retention period of 15 days.

```
PNC
MARTIN,JC999,NT1.
PFLIST,PREFIX=$COMMON PREFIX$,U=OUT,VRN,NT=UP1000=UP1001.
PFLOAD,I=OUT.
6/7/8/9
```

Example 2:

All cycles of the file "MONEY$PF" located on two separate sets of 7-track PF dump tapes UP0001-UP0002 and UP0009-UP00010 are reloaded on disk with their original retention periods.

```
PNC
JAMES,JC999,MT1.
PFLOAD,PFN=$MONEY$$PF$,CY=ALL,RP=SAME,
MT=UP0001=UP0002,MT=UP0009=UP0010.
6/7/8/9
```

Example 3:

In this example, all files stored on the 7-track PF dump tape UP1020 and the 9-track PF dump tape UP1040 are reloaded with an infinite retention period. Files already on disk are ignored.

```
PNC
SMITH,JC999,MT1,NT1.
PFLOAD,ALL,MT=UP1020,NT=UP1040,RP=999,DUP=IGNORE.
6/7/8/9
```

Example 4:

In this example, the first cycle of JONESLGO is reloaded from the global tape set, UP0012 and UP0034. All cycles of SMITHLGO are reloaded from the 9-track dump tape UP2109. Cycle 60 of GOODPF is reloaded from the 7-track tape UP2001 (unless a copy already exists on disk). Cycle 1 of BADPF is reloaded from the 9-track tape UP1984 with its original retention period. All PFs with the exception of BADPF will have a retention period of 60 days.

```
PNC
LASTONE,JC999,MT1,NT1.
PFLOAD,MT=UP0012,NT=UP0034,RP=60,DUP=NEWNAME.
7/8/9
JONESLGO
SMITHLGO,CY=ALL,NT=UP2109
GOODPF,CY=60,MT=UP2001,DUP=IGNORE
BADPF,CY=1,NT=UP1984,RP=SAME
6/7/8/9
```

6
Magnetic Tapes

6.1
Reader's Guide and Glossary

This chapter is intended to be a comprehensive guide to magnetic tape usage under SCOPE/HUSTLER. Sections 6.2 through 6.6 give the user the basic information necessary to use tapes, including a discussion of tape vs. disk storage (6.2), information on obtaining tapes (6.3), appropriate control statements for reserving tape drives (6.4) and requesting tapes (6.5), and a description of tape error messages (6.6). The interested user may also want to read Sections 6.7 and 6.8, which give detailed descriptions of tape equipment and tape structures.

Sections 6.9 through 6.16 will be useful to the more experienced user. The topics covered are: the structure of data on tapes (6.9 and 6.10), tape labels (6.11 and 6.12), tape security (6.13), tape parity error procedures (6.14), and special problems involving tapes (6.15). COMPASS procedures have been separated from the text and placed in Section 6.16; this was done to facilitate their removal when the *SCOPE/HUSTLER COMPASS Reference Manual* is completed.

Several of the terms used in this chapter to describe files have been redefined with the purpose of eliminating overlapping terminology and making the terms more precise. These terms are defined below.

File
A file is a physically connected set of information. It is all the data between the beginning-of-information and the end-of-information.

Record or Unit Record
A record is a physical division of data comparable to a punched card, or one line of code or string of characters. Different types of records may be defined under Cyber Record Manager.

Logical Record
A logical record has been redefined as a group of logically related data within a file. A logical record comprises one or more unit records but **has no unique delimiter**. For example, data for a single observation may span several cards. This group of cards is termed a "logical record"; it is not separated from other cards by a special delimiter.

Section
A section consists of one or more records. It is less than a partition and greater than a record, but it may be identical to either or both. A section begins at beginning-of-information or immediately after the end of the preceding section, and ends when an end-of-section (EOS) delimiter is detected.

End-of-Section Delimiter (EOS)
All sections are terminated by an end-of-section delimiter. This delimiter is automatically appended to the user's data upon writing a section but is not returned with the data when reading a section. A level number is associated with each end-of-section delimiter. This level number may be from 0 to 17_8 (octal). A level 17_8 end-of-section is called an end-of-partition delimiter.

Partition
A partition consists of one or more sections terminated by an end-of-partition delimiter.

End-of-Partition (EOP)
An EOP is equivalent to an end-of-section delimiter of level 17_8.

Block and Physical Record
A block is a physical grouping of information on tape. It is the smallest amount of information which may be written or read on a magnetic tape, although block size may vary. Blocks are delimited by interblock gaps.

Although sometimes used to describe disk files, the term block has little significance for those files. On disk the smallest unit of information read or written is fixed in size and is called a sector or a physical record (PRU). When the term block is applied to disk files it is used as a logical grouping of information physically recorded as a whole number multiple of PRUs.

Volume
A volume is the unit of information contained on a physical reel of magnetic tape. A file may consist of more than one volume, and a volume can contain any number of sections or partitions.

6.2 Tape vs. Disk Storage

Magnetic tapes are an economical alternative to permanent files for storing large or infrequently used files. They are also a compact medium for transporting large amounts of information from one computer to another.

This section discusses some of the differences between magnetic tape storage and disk storage, which most users are more familiar with. These comparisons will also furnish a basis for choosing the type of storage best suited to the user's needs.

6.2.1 Economic Advantages

The chief attraction of magnetic tape is that it is the least expensive medium for storing large or infrequently used files. For example, at the current disk storage rate of about 1 cent per 80 column card image per month, it costs $20 per month (about $.67 a day) to maintain a 4000 card (320,000 character) permanent file. In many cases it is wiser to invest a flat fee covering purchase price, storage and handling fees for a 2400 foot magnetic tape and thereafter pay only tape read charges and a 25-cent tape mount charge each time the tape is used.

A coded 7-track tape written at 800 characters per inch (cpi) can hold the equivalent of 15 million characters (24,000 disk PRUs); a binary tape can hold 20 million characters (31,000 disk PRUs). A coded 9-track tape written at 1600 cpi can hold 23 million characters (36,000 disk PRUs); a binary tape can hold as much as 46 million characters (72,000 PRUs). Several unrelated sets of data can be stored on one tape, as in the case of PFDUMP, thus allowing for maximum use of the entire tape.

Another factor is the frequency with which a file is modified. The disk is a random-access device which allows a user to modify one section without affecting any of the others. Magnetic tape is a sequential device, which means that any change to a given block will affect all of the blocks which follow it. Usually the user must rewrite the entire tape in order to modify one of its blocks. A two-tape process is recommended to do this; see Section 6.2.6.

Another critical factor, in addition to size and frequency of use, is whether or not the user's problem number is subsidized. If it is, the user's department will be billed for only a fraction of the permanent file costs but for all of the tape mount costs. However, the department is billed for only a fraction of the charges for channel time (roughly speaking, the time the tape is in motion).

Punched cards, incidentally, are the most expensive form of storage except for very small or seldom used files. A file of 320,000 characters is equivalent to at least 4000 coded cards, and usually more. It costs $4.32 to read a deck of 4000 cards, not to mention the burden of carrying those two 5-pound boxes.

6.2.2 Random Access vs. Sequential Access

A random access, or direct access, device is one where storage is organized into addressable locations and where the access time to any location is constant, relatively short, and independent of the location previously addressed. A sequential access device is one where data can only be accessed in the sequence in which it is stored on the device; in other words, to process a particular item, one must first process or pass over all of the preceding items.

The best example of random access storage is magnetic core memory. In comparison, magnetic disks provide only "semi-random" access, for even though disk surfaces are organized into addressable sectors, the access time to a given sector can vary considerably depending on whether the read/write head assembly must be repositioned. Occasionally magnetic tape is formatted so that it too can provide semi-random access. But since the maximum access time—the time necessary to reposition the tape from a section at the beginning to a section at the end—is measured in minutes rather than milliseconds, random access tapes are simply too slow for high-speed computers. The 6500 and all comparable computers use magnetic tape strictly as a sequential access device.

To illustrate some differences between tape and disk structure, think of the disk as a book. Like the pages of a book, each disk PRU has a fixed capacity and a specific location designated by a number (i.e., an address). A file, however, would not necessarily be written on successive pages. For instance, it might be written on pages 40, 41, 30, 31, and 52 in that order. SCOPE/HUSTLER maintains the sequence of the file by means of an index listing the "page numbers" in proper order. Indexing enables SCOPE/HUSTLER to add, delete, or rewrite pages without affecting any of the other files. It also enables SCOPE/HUSTLER to retrieve an item without examining the data itself. In effect, SCOPE/HUSTLER looks up the item in a table of contents and then flips to the indicated page number.

In contrast, magnetic tape is similar in both appearance and logical structure to a scroll. Data is written in variable-length blocks which have no fixed location on the tape. As a result, the only way to locate a particular item is to start at the beginning of the tape and read each block.

6.2.3 Convenience

The difference between random access and sequential access accounts for much of the added convenience of disk files. One immediate benefit of using disk files is better turnaround time. Since random access storage can be allocated to several jobs at one time, there is no waiting for access to a disk file. Jobs requesting a magnetic tape, on the other hand, must wait for an available tape drive. If the job uses several tapes simultaneously, or if the system is heavily loaded with tape requests, the job may have to wait several hours.

Another convenience of disk files is that the system takes care of all bookkeeping chores, while the magnetic tape user must make careful records of which reel contains which partitions and in what order the partitions are written. If the user fails to do this or makes a mistake, valued information may be accidentally overwritten.

6.2.4 Writing and Rewriting Data

When information is being recorded on magnetic tape, an erase head removes previously recorded data before the tape moves past the write head (see Section 6.7.2). Information beyond the overwritten portion of tape, although not erased, must also be considered destroyed. This is also true when a write function is used to modify a disk file; the rest of the file is dropped and the last PRU written becomes the end-of-information. But SCOPE/HUSTLER also provides special rewrite-in-place functions for disk files, available as COMPASS macros, which allow the user to replace a section in the middle of a disk file with another section of the same length.

While it may seem possible to do a rewrite-in-place on a magnetic tape, this procedure (if it were permitted) would be unreliable for the following reasons:

1. Tape motion is not sufficiently consistent to guarantee that two blocks containing the same number of characters will occupy the same length of tape.

2. The erase head, positioned about 1/2 inch ahead of the write heads, might erase part of the next block.

3. If a parity error occurred during the write, the tape drive would erase about six inches of tape before retrying the write.

SCOPE/HUSTLER does not permit the user to issue a read following a write, without first repositioning the file. Repositioning a file causes SCOPE/HUSTLER to write an end-of-information indication which, on a tape, would be written over previously recorded data.

6.2.5 Security

There are two kinds of security concerns: safety and privacy.

In principle, a permanent file will only be destroyed if the expiration date has passed, the owner's PN is expired, or the owner's dollar balance is expended. In reality, though, the safety of a permanent file is obviously tied to the reliability of the computer and its operating system. Recognizing this, the Computer Laboratory maintains a temporary backup copy on magnetic tape for every permanent file, including the expired files purged by the Computer Laboratory (see Section 5.4).

The use of magnetic tape to make backup copies of permanent files should not be interpreted to mean that magnetic tape is inherently safer than a permanent file. Tapes, more than disks, are subject to the dangers of human handling; they can be dropped, they can be stretched by malfunctions of the tape drive, and they can be permanently damaged if stored under improper conditions. In rare cases, a mechanical failure of the tape drive will stretch or snap a tape in two. A tape will also wear out if used frequently enough.

To protect a file that would be expensive or impossible to recreate, the user should keep a duplicate on a separate tape and use that copy only as a backup tape (see Section 6.2.6). If a tape is removed from the machine room, it should be protected from dust and kept in an environment free from extremes of temperature and humidity. Tapes should be stored on edge, not laid flat on top of one another.

When appropriate precautions are taken, magnetic tape provides a high degree of reliability. Permanent files, however, provide a higher and more flexible level of privacy. Access to a permanent file may be controlled by a set of up to five passwords, each protecting a different type of access (read, extend, modify, control, and turnkey).

The utilities PFDUMP and PFLOAD described in Chapter 5 can be used to back up files with the full security of permanent files. Other methods of backing up files which use the SCOPE/HUSTLER automatic tape labeling feature restrict permission to write on a magnetic tape to jobs submitted with the owner's problem number. However, anyone can request and read a non-PFDUMP tape, unless the owner removes it from the machine room. See Section 6.13 for a more complete discussion of the security system.

6.2.6 Backup of Important Data

For safety reasons, the Computer Laboratory strongly recommends that a two tape system be used, with one tape acting as the "working" copy of the data, and the other as a "backup" copy. This backup system should also be used for multi-volume tape sets.

Whenever the user wishes to change the working copy, the modified data is written onto the backup copy, leaving the working copy unchanged. Once the desired changes are made and verified to be correct, the new copy then becomes the working copy. The old working copy becomes the new backup copy.

Each time a change is made, this process is repeated, saving the original copy of the data in case errors occur in the updating process. Note: If a tape is infrequently modified (i.e., less than once a year), the backup copy should be exercised periodically to prevent deterioration of the tape (see Section 6.3.2).

6.3 Using Tapes at MSU

This section describes the procedures necessary to obtain and use magnetic tapes on the MSU 6500.

6.3.1 Obtaining a Tape

Magnetic tapes can either be purchased from the Computer Laboratory or obtained at another installation. The Computer Laboratory sells a high-quality tape in the 2400-foot length; also available are 200-foot, 400-foot, and 600-foot tapes for use in mailing data to and from other installations.

Tapes are handled only by the machine room operators; therefore, a tape to be used on the 6500 must be stored in the machine room.

All requests for purchase of tapes are made via a Tape Service Request Form obtainable at the Service Window in Room 208 Computer Center. Policies regarding these services are discussed in the *Facilities and Policies Handbook*, Section 7.3.

6.3.2 Storage of Tapes

Magnetic tapes used with the 6500 must be stored in the Computer Laboratory's tape library. There are two types of storage: temporary and permanent.

Temporary storage allows a user to store a tape in the tape library for a short period of time (normally ten working days). It is designed to allow a tape to be used on a short-term project and then removed. Note: A user who has a labeled tape in temporary storage may encounter difficulties when requesting the tape. See Section 6.15.2 under "Incompatible Labels" for more information.

A tape that the user wishes to store indefinitely will be assigned to permanent storage. Users who bring tapes from other sites pay a one-time storage fee. If a tape is purchased from the Computer Laboratory, the storage fee is included in the purchase price. Tapes are stored in a controlled environment and exercised periodically.

The Computer Laboratory offers several additional tape services, as described below. Requests for these services can be made at the Service Window in Room 208. There is a nominal charge for each service.

1. Cleaning a tape

2. Relabeling a tape (see Section 6.13.2)

3. Stripping and adding load points (i.e. removing the first several feet of a tape and adding a beginning-of-tape reflective marker. This is done when the first part of a tape is bad, and the remaining length of the tape is considered to be good.)

6.3.3 Visual Reel Name (VRN)

When a tape is received for storage in the tape library, it is given a visual reel name (VRN), which is written on a paper label and glued to the tape reel. The VRN acts as an identifier for both the user and the operator.

To request a tape, the user's job must contain a REQUEST control statement (see Section 6.5.1), the COMPASS macro MSUREQ (see Section 6.16.1), or the FORTRAN function MSUREQ (see Section 6.5.2), specifying the VRN of the desired tape. Such a request causes a message to be displayed on the operator's console, requesting that the tape be mounted and assigned.

Tapes that have a VRN prefix 'VIM' are usable only by Computer Laboratory personnel. Tapes with a VRN prefix 'PF' may only be written on by Computer Laboratory personnel but may be read by anyone using PFLIST or PFLOAD. VRNs with the prefix 'UP' are used only with the utilities PFLIST, PFDUMP and PFLOAD. Further information on PFLIST, PFDUMP and PFLOAD may be found in Section 5.4.

6.4 Tape Drive Reservation

To permit job scheduling and efficient allocation of the 7-track and 9-track tape drives SCOPE/HUSTLER requires the user to specify the maximum number of tape drives that will be used simultaneously by the job. The system does not display a tape request to the operator unless there are enough tape drives available (unreserved) to satisfy the maximum tape drive requirement of that job. When the first tape is assigned to the job, the system reserves the remainder of its tape drive requirement so that the tape drives will be immediately available when needed.

The job 7-track tape drive reservation is set initially by the job card 'MTmt' parameter, which has a default value of 0. The 9-track drive reservation is set by the 'NTnt' parameter on the job card, which also has a default of 0. These values may be reset using the TAPRES control statement, the MT=mt and NT=nt parameters of the RETURN control statement, or one of the TAPRES object-time requests. If a job subsequently requests more tape drives than it has reserved, it will be aborted. In addition, all tape reservation changes, whether specified by a TAPRES statement, a RETURN statement, or an object-time request, are subject to the following rules.

1. A job cannot raise its tape reservation if tapes are currently assigned to it.

2. A job cannot lower its tape reservation below the number of tapes currently assigned to it.

3. A maximum of four 7-track and four 9-track tape drives may be reserved. The minimum reservation is zero.

4. A job cannot raise its tape reservation above that specified on the job card.

6.4.1 Job Card

If a magnetic tape is used in a job, a tape drive reservation parameter must be included on the job card. If this parameter does not appear on the job card, the job will abort at the first attempt to use the tape; the dayfile will contain the message

```
MORE TAPES REQUESTED THAN RESERVED.
```

Seven-track and 9-track tape drives are reserved separately. The parameters are:

MTmt the number of 7-track tape drives to be reserved for the job; $0 \leq mt \leq 4$. The default is 0; that is, no 7-track drive will be reserved if this parameter is omitted.

NTnt the number of 9-track tape drives reserved; $0 \leq nt \leq 4$. The default is 0.

6.4.2 TAPRES Control Statement

The TAPRES control statement specifies the maximum number of tape drives needed by the job at any one time. Its use is subject to the rules listed in Section 6.4.

```
TAPRES,MT=mt,NT=nt.
```

MT=mt the number of 7-track tape drives ($0 \leq mt \leq 4$) to be reserved when the first tape is assigned.

NT=nt the number of 9-track tape drives ($0 \leq nt \leq 4$) to be reserved when the first tape is assigned.

Note: mt and nt may be the word 'SAME' in which case the corresponding reservation is unchanged.

Example:

```
PNC
id,MT3,NT1.
PW=password
TAPRES,MT=1,NT=0.
REQUEST,OLDPL,VRN=502.
UPDATE.
RETURN,OLDPL.                  Tape 502 must be returned (see Section 6.4.3), giving
                               a tape reservation of 0, before TAPRES can raise the
                               7-track reservation to 3.
COBOL,I=COMPILE.
TAPRES,MT=3,NT=1.
REQUEST,TAPE1,VRN=1021.
REQUEST,TAPE2,VRN=1022.
REQUEST,TAPE3,VRN=1023.
REQUEST,TAPE10,VRN=989,NT,RW.
LGO.
7/8/9

UPDATE directives

6/7/8/9
```

This example illustrates an intelligent use of the TAPRES statement, since it allows the user to retrieve, update, compile and load the program before reserving all four tape drives. If update or compilation errors occur, the user will get relatively fast turnaround. On the other hand, if the user had not included the TAPRES statement to decrease the tape reservation count, he/she would have had to wait until three 7-track drives and one 9-track drive were available before performing the UPDATE. The system does not actually reserve a tape drive for the job until the first use of a tape (see Section 6.5) and so, although the REQUEST statements for tape 1021-1023 and 989 are placed before the LGO statement, the drives are not reserved and the tapes are not mounted until the files are used by the program.

Suppose the program loads successfully and references tape 1022. The job must then wait until four tape drives are free before the operator can mount and assign tape 1022. When 1022 is assigned, SCOPE/HUSTLER reserves all four tape drives for the job.

Note that the tape reservation count specified on a TAPRES statement may never exceed the number of 7- and 9-track tape drives specified on the job card. Note also that the RETURN control statement normally decrements the tape reservation count (see Section 6.4.3) and so it is often necessary to reset the reservation counts after a RETURN.

In the example, the tape reservation counts for 7- and 9-track tapes are zero after the RETURN statement. If more tape drives are needed after a RETURN, the reservation count may be reset with a TAPRES statement or with optional parameters on the RETURN statement (see Section 6.4.3).

6.4.3 RETURN Control Statement

The RETURN control statement detaches the specified local files from the user's job and returns them to the system for final disposition. The system handling of a returned file depends on whether the file is a cataloged permanent file, a special disposition file (print, punch, or input), or a magnetic tape file. If it is a tape file, the tape is rewound and unloaded, and the tape drive is made available to other jobs.

Normally the job tape reservation is decremented by 1 for each tape returned. To avoid decrementing the tape reservation, the user may specify a tape reservation parameter.

RETURN,lfn$_1$,lfn$_2$,...[,MT=mt][,NT=nt].

lfn the name(s) of the local file(s) being returned.

MT=mt either a digit (0≤mt≤4) specifying a new 7-track tape reservation, or SAME specifying that the 7-track tape reservation is to remain the same.

NT=nt either a digit (0≤nt≤4) specifying a new 9-track tape reservation, or SAME specifying that the 9-track tape reservation is to remain the same.

The values of mt and nt are subject to rules 1-4 listed at the beginning of Section 6.4. For example, suppose that a job has reserved three 7-track tape drives and that all three tapes are assigned. If only one tape is returned (leaving two still assigned), the MT parameter cannot be used to lower the reservation to 1 or to raise it to 4. If all three tapes are returned, mt may then be assigned any value from 1 to 3.

Example 1: Raising the tape unit reservation.

In the preceding section (illustrating the use of the TAPRES control statement) the same effect could have been achieved by using the RETURN statement MT parameter when tape 502 was returned:

```
RETURN,OLDPL,MT=3,NT=1.
```

Example 2: Maintaining the current tape drive reservation.

```
PNC
id,JC100,NT2.
REQUEST,TAPE1,VRN=100,RW,NT.
REQUEST,TAPE2,VRN=101,NT.
COPYCR,TAPE2,TAPE1.
RETURN,TAPE2,NT=SAME.
REQUEST,TAPE3,VRN=102,NT.
COPYCR,TAPE3,TAPE1.
6/7/8/9
```

If the NT parameter were omitted from the RETURN statement, the job tape drive reservation would drop to 1 when tape 101 is returned, and the job would abort on the request for tape 102.

6.5 Tape Requests

Magnetic tape files must be explicitly requested before they are used by a job. Tapes are usually requested with a REQUEST control statement, which is described in Section 6.5.1. COMPASS programmers may make object-time tape requests using the MSUREQ macro, which is described in Section 6.16.1.

The user's tape request is relayed to the operator by a message displayed on the console. Although the tape request must be made before the file is referenced, the request message is not displayed until a program actually attempts to use the file. This procedure, known as "first use assignment," avoids having the operator mount tapes that are never used because of fatal job errors. When a tape file is first referenced by a program, the job is swapped out and placed in a wait state until the operator has assigned the tape. As described in Section 6.4, the operator cannot assign the tape unless there are enough tape drives available to meet the job's maximum tape drive requirement. Thus, it may take from several minutes to several hours to fulfill a tape request, depending on the job's tape drive reservation count and the number of other jobs awaiting tape assignments.

The tape request message displays the tape VRN and indicates whether the tape is to be mounted with or without a write-enable ring (see Section 6.7.3). After mounting the reel and positioning the tape, the tape is assigned to the user's job. If the tape is labeled (see Section 6.11), it is assigned automatically by the system; if the tape is unlabeled, the operator assigns the tape by typing a command on the console keyboard. At this point the system initiates the tape security checks described in Section 6.13.1. The tape security system will reject the assignment if the ring status is incorrect, if the VRN is incorrect, or if the problem number is incorrect when a ring (see Section 6.7.3) is requested. Additional label checking is performed on tapes requested as labeled (see Section 6.11.5), but these do not provide any real security since the operator normally instructs the system to continue processing the job in spite of label discrepancies.

6.5.1 REQUEST Control Statement

The REQUEST control statement is used to request the assignment of a device for either a disk or magnetic tape file. Because SCOPE/HUSTLER will create a disk file automatically whenever a job references a previously undefined local file name, explicit requests for disk files are not necessary.

REQUEST,lfn[,VRN=vrn_1=vrn_2=...][,NEWPN=pn][,NB][,rng]
[,fmt][,lbl][,dns][,cm][,NR].

lfn the local file name to be assigned to the file. If lfn is the only parameter specified, the request is assumed to be for a disk file; otherwise it is assumed to be a tape request.

VRN=vrn a list of visual reel names (VRN). When the volume trailer label is encountered, the system will rewind and unload the reel and request the next VRN in the list. If the VRN list is exhausted, the job will abort. Normal termination will occur only if the system encounters a file trailer label (EOF1). The user may specify up to 62 VRNS in this list.

If the VRN list is omitted but some other tape parameter is specified, a scratch tape will be assigned. **Caution:** In the interest of efficiency, disk files should be used rather than scratch tapes. There is no way to guarantee the contents of a scratch tape from one run to another, or even to request the same scratch tape.

NEWPN=pn specifies a 6 to 7 character problem number, or zero. This parameter is used only when creating a labeled tape; it has no effect when either the RO or Z parameter is specified. The value specified by pn is written into the PN field of the header label. A value of 0 (zero) will permit any user to write on the tape.

If NEWPN=pn is omitted when a labeled tape is created, the problem number from the job PNC is used, unless the tape is a scratch tape, in which case the PN field is left blank.

NB no brackets—suppresses noise bracketing of unusable tape; alternate procedures for recovering write parity errors are used instead. This parameter does not apply to 9-track tapes. See Section 6.14.

rng specifies whether the reel is to be mounted with or without a write-enable ring (see Section 6.7.3):

RW	read-write (mount with ring)
RO	read-only (mount without a ring)
omitted	read-only (mount without a ring)

fmt specifies the data format (see Section 6.9):

S	stranger tape
L	long record stranger tape
omitted	SCOPE tape

lbl specifies the label format (see Section 6.11):

Z	unlabeled
Y	MSU 3600 labels (obsolete)
NS	nonstandard labels; may be used in conjunction with Z.
omitted	SCOPE-ANSI labels

If an unlabeled tape is requested as labeled and the first action is a write, a labeled tape will be created. But if the first action is a read, the system will attempt to check label information against the values of the FET and, finding unrecognizable information, will post the "TYPE DROP OR GO" message to the operator. The same action will occur if the wrong label style is specified. If a labeled tape is requested as unlabeled, (Z parameter) the tape will be positioned past the label. If both Z and NS are specified, the tape will be positioned before the label (at load point) to allow user processing of labels.

dns specifies the density at which the data will be recorded or read (see Section 6.8.3). This also specifies whether the tape is 7-track or 9-track. The density must always be specified on a request for a 9-track SCOPE tape; otherwise the system assumes the tape is 7-track. If the tape is a 9-track stranger tape, the cm parameter described below takes precedence. When a labeled tape is read, the density parameter is overridden by the density specified in the header labels, if the two values differ.

7-track tapes		9-track tapes	
LO	200 cpi	HD	800 cpi
HI	556 cpi	NT	1600 cpi (phase encoded)
HY	800 cpi	PE	1600 cpi (phase encoded)
MT	556 cpi		
omitted	556 cpi		

cm character conversion mode for 9-track stranger (S and L) tapes (see Section 6.9.3). Inclusion of this parameter implies that the tape is 9-track. The density is assumed to be 1600 cpi (phase encoded); if an appropriate preamble and postamble for a phase encoded tape (see Section 6.8.7) are not found, the tape will be reread at 800 cpi.

AS	perform ASCII to Display code character translation.
EB	perform EBCDIC to Display code character translation.
omitted	perform ASCII to Display code character translation.

NR no parity error recovery; blocks are returned to the buffer exactly as they are read, and no parity error checking is performed (see Section 6.14).

When a user reads a 9-track S or L tape in coded mode, the tape drive control unit converts the data from 8-bit ASCII or EBCDIC to a 6-bit Display code value. The opposite conversion is performed on output, using a 64-character subset of the ASCII or EBCDIC character set. See Appendix A for the conversion table.

Examples:

1. REQUEST,TAPE1,VRN=1025,RW,HY.

This statement requests tape 1025 with a write-enable ring and assigns it the local file name TAPE1. The tape will be written in SCOPE format with SCOPE-ANSI labels at a density of 800 cpi.

2. REQUEST,TAPE2,VRN=1025.

This statement would be used to request the tape created in Example 1 if the user only intends to read it. The tape will be mounted without a ring. Note that the HY parameter is omitted since the density is specified in the SCOPE volume header label.

3. REQUEST,TAPE3,VRN=601=602=603.

This statement requests tape 601, with continuation reels 602 and 603 as SCOPE-ANSI labeled, SCOPE-formatted tapes. The reels will be mounted without a ring.

4. REQUEST,TAPE4,VRN=3789,S,Z.

This statement requests tape 3789 as a HI density, unlabeled stranger tape. The tape will be mounted without a ring.

5. REQUEST,TAPE5,VRN=102,RW,NEWPN=016930.

This statement requests tape 102, a labeled SCOPE tape, with a ring. If the first action is a write, 016930 will be written into the PN field of the volume header label.

6. REQUEST,TAPE6,RW.

This statement requests a scratch tape.

7. REQUEST,TAPE7,VRN=107,EB.

This statement requests tape 107, a 9-track stranger tape coded in EBCDIC.

6.5.2 MSUREQ Subroutine

A function MSUREQ is available to request tapes from FORTRAN. COBOL users can also utilize this function. MSUREQ is documented in the Computer Laboratory Publication *FORTRAN Extended Library Routines*.

6.6 Tape Error Messages

When an error occurs in reading or writing a tape, the system routines attempt to diagnose the error and provide an informative dayfile message. Because of space limitations in some system tape-handling programs, these dayfile messages are sometimes quite cryptic. The common abbreviations used are:

RD	read
WRT	write
RVD	recovered
PAR	parity
ERR	error
REC	record
DEV CAP EXC	device capacity exceeded
SNR	standard noise record
IRG	inter-record (interblock) gap

Thus, the dayfile message "RD RVD LOST DATA" means that lost data was recovered during a read operation. "RD ERR TAPE PAR ERR" means an unrecoverable tape parity error occurred during a read operation.

Since the complete list of error messages is lengthy and changes frequently, it is not included in this manual. Users who encounter a dayfile message whose meaning is not clear should contact the consultants in the User Information Center (Room 313 Computer Center, phone 353-1800) for help.

6.7 Tape Equipment

This section discusses the physical characteristics of tapes and tape drives.

6.7.1 Magnetic Tape

Magnetic tape consists of a long strip of plastic, called the base or backing, which is coated on one side with a highly retentive magnetic material. Generally this coating is composed of a fine iron oxide powder mixed into a binder of organic resins. Information is represented in this medium by alternating the polarity of minute magnetized areas.

The standard width for magnetic computer tape is 1/2 inch. Length and reel size vary, but the standard full size reel measures 10-1/2 inches in diameter and holds about 2400 feet of tape. Because the Computer Laboratory stores tape reels in seals rather than canisters, the reel should be solid-sided to protect the tape from dust and dents. A tape seal is a plastic ring which snaps around the outer edge of the reel, and which provides a hook for hanging the reel on a storage rack.

The quality and price of magnetic tape can vary substantially. For top performance, the oxide particles must be a uniform size and shape so that the coating is not abrasive. The oxide layer must also be a uniform thickness, or there will be sharp differences in the strength of the recorded signal. The plastic backing must be straight, fairly strong, and free of splices. The substance binding the oxide to the backing must be tough but flexible so that the oxide does not chip or flake off. It must also be non-adhesive so that tape layers do not stick to one another when wound tightly on a reel. When purchasing a tape, remember that the differences in price between a high-grade tape and an ordinary tape seldom covers the cost of replacing lost data.

6.7.2 Tape Drives and Control Units

The device that reads and writes magnetic tape is called a tape drive, tape unit or tape transport. It consists of a set of read/write heads and a mechanism for transporting the tape past the heads, from the supply reel to a take-up reel. Basically the tape drive resembles a common audio recorder but has the ability to drive a tape at high rates of speed (150 inches/second for 7-track drives and 200 inches/second for 9-track drives) and to stop and restart the tape motion in distances of less than an inch.

MSU's 6500 system is equipped with 7-track tape drives and 9-track tape drives. The 7-track drives and the 9-track drives are connected to separate control units, or controllers. Both control units buffer and control the flow of the information between the tape drives and two of the twelve 6500 data channels. All of the read and write logic (circuitry) is contained in the control units rather than the drives.

The head assembly of a tape drive consists of individual read and write heads, an erase head, two tape cleaners, and a pneumatic pressure pad. Each of the read/write heads has two magnetic gaps: one for writing and one for reading. The gaps are arranged so that during a write operation, the tape first passes under the write gap to record the data and then under the read gap to check the writing. The broad band erase head removes any information recorded on the tape before new information is recorded at the write gap. Precise contact pressure between the tape and the head gaps is maintained by air blown through the pneumatic pad. The two tape cleaners, located on either side of the heads, vacuum loose particles from the tape surface.

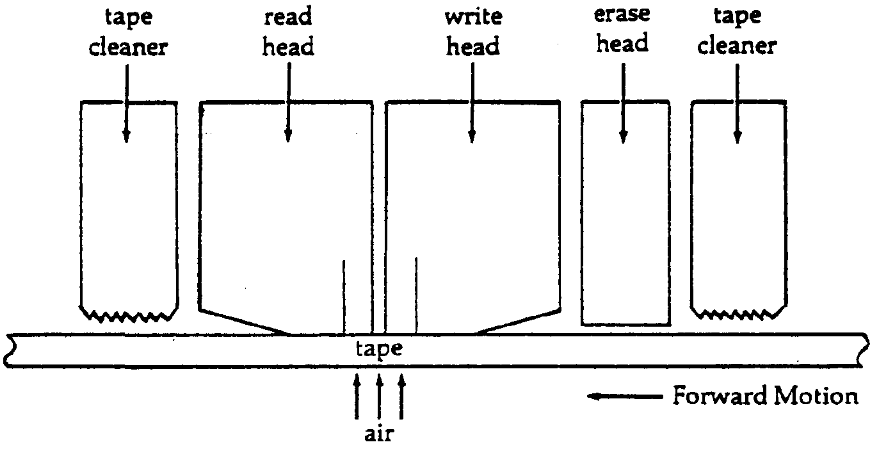


Figure 6-1: Read-Write Unit

A write head is an electromagnet consisting of a coil of wire wound around a steel core. To concentrate the magnetic field that is induced when current is sent through the coil, the core is U-shaped, leaving only a small gap between the poles. When a magnetic tape is recorded, the oxide particles directly beneath the write gap are magnetized by this field. The alignment (or polarity) of the magnetized oxide particles is determined by the polarity of the gap, which in turn is determined by the direction of electrical flow through the coil. Thus one can reverse the polarity of the oxide particles by reversing the electrical flow. Such a reversal is called a **flux change**.

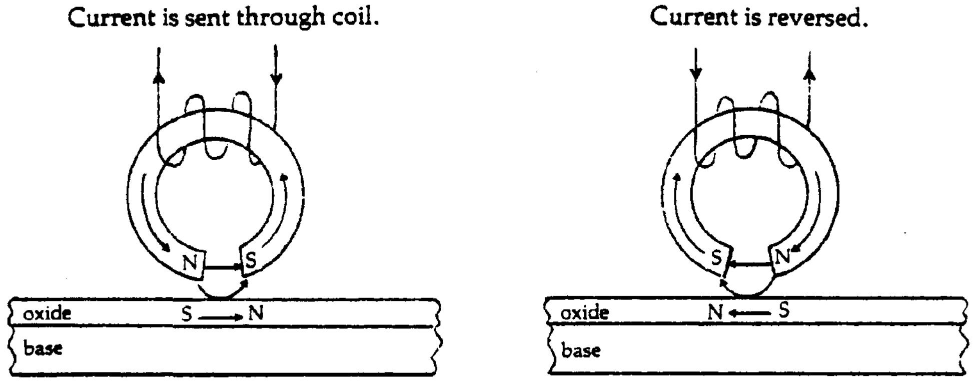


Figure 6-2: Write Head Flux Change

A read head is similar in construction to a write head, except that the magnetic field of the oxide particles is used to induce a current in the coil. When a read head senses a flux change, there is not only a reversal of electrical flow in the coil, but a surge of voltage which the read circuits are designed to detect and amplify.

6.7.3
Write-Enable Ring

Magnetic tape reels are supplied with a plastic ring that fits into a groove on the backside of the reel. When the reel is mounted on a tape drive, the ring presses a switch that allows the drive to write. If the ring is removed from the reel, the tape can only be read.

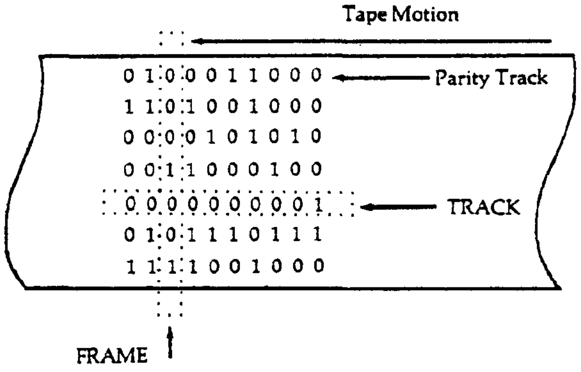
Under SCOPE/HUSTLER the user's tape request always indicates whether a reel is to be mounted with or without the write-enable ring (see Section 6.5.1). If the operator fails to comply with this part of the request, the system will reject the tape assignment and instruct the operator to insert or remove the ring, as requested.

6.8
Physical Data Structure

This section describes how information is represented on a tape.

6.8.1
Tracks and Frames

One may visualize the information recorded on a magnetic tape as a matrix of magnetized spots, each spot representing a bit. The rows of bits that run longitudinally parallel to the edges of the tape are called **tracks**, and the columns are called **frames**. The 7-track tape drives have read/write heads that simultaneously process seven tracks of information perpendicular to the tape edges. A frame, therefore, contains seven bits, of which six are **data bits** and one is a **parity bit** (see Section 6.8.7). Since 7-track coded tapes consist of 6-bit codes, written one character per frame, the terms "frame" and "character" are often used interchangeably. The 9-track tape drives are similar; however, their read/write heads are made to process nine tracks of information. Each frame then contains eight data bits and one parity bit.



```
0 1 0 0 0 1 1 0 0 0
1 1 0 1 0 0 1 0 0 0
0 0 0 0 1 0 1 0 1 0
0 0 1 1 0 0 0 1 0 0
0 0 0 0 0 0 0 0 0 1
0 1 0 1 1 1 0 1 1 1
1 1 1 1 0 0 1 0 0 0
```

Figure 6-3: Frames and Tracks

It should be clear that the difference between a "7-track tape" and a "9-track tape" is not a property of the tapes but of the tape drives that recorded the data. Seven-track tape drives were once the industry standard, but they are being replaced by 9-track drives due to the popularity of the IBM System/360 computers and the EBCDIC character set. Tapes written on a 9-track drive cannot be read on a 7-track drive and vice versa (see Section 6.15.1).

6.8.2 Recording Techniques

NRZI

NRZI (which stands for Non-Return to Zero—Invert on ones) is the most commonly used recording method for 7-track tapes and for 9-track tapes recorded at 800 cpi (see Section 6.8.3).

The NRZI technique explicitly records only the "1" bits; that is, a flux change (see Section 6.7.2) indicates a "1" bit and the absence of a flux change indicates a "0" bit. At this point you can probably see that the image of magnetically encoded data as a "matrix of magnetized spots" given in Section 6.8.1 is not entirely accurate. Each track is a continuous stream of fully magnetized particles. The demarcation of a frame, therefore, is a matter of timing rather than a physical gap.

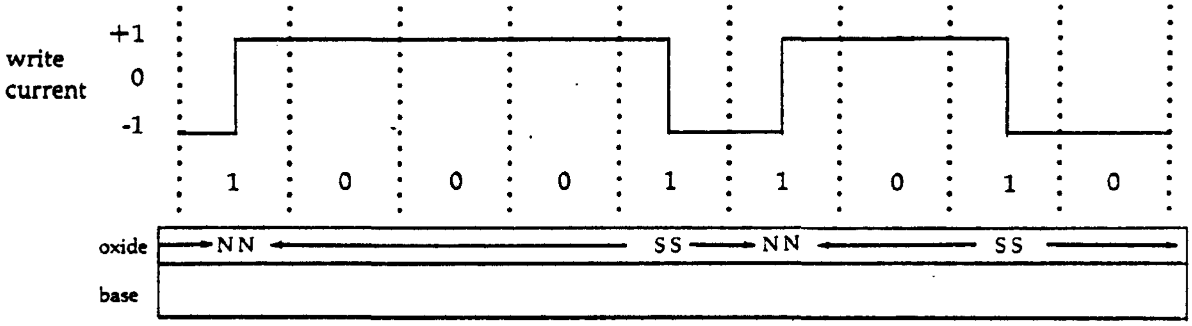


Figure 6-4: NRZI (Non Return to Zero—Invert on ones)

Phase Encoding

Another technique, known as phase encoding (PE), is commonly used for very high density (1600 cpi) 9-track tapes. In the PE technique, a "1" bit is recorded by sending current to the write head in one direction for the first half of the bit interval, and then in the other direction for the second half. A "0" bit is recorded in the same way except that the phase is shifted 180 degrees. Thus at the center of each bit interval there is a positive voltage for a "1" bit or a negative voltage for a "0" bit.

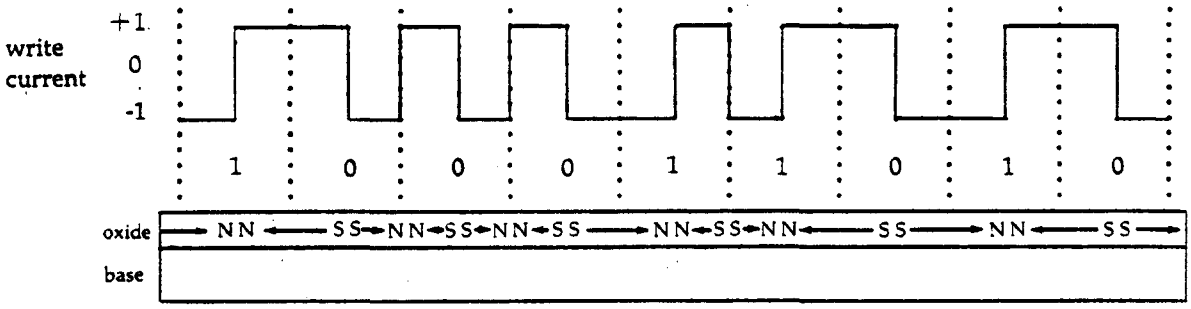


Figure 6-5: Phase Encoding

6.8.3 Density

Density refers to the spacing between successive tape frames, and it is measured in terms of the number of characters per inch (cpi) in each track. (It is sometimes referred to as bits per inch, bpi, or frames per inch, fpi.) For instance, a density of 800 cpi means that data is recorded at 800 characters (or frames) per inch. The CDC 7-track tape drives can read or write at any of three densities: 200 cpi (LO), 556 cpi (HI), or 800 cpi (HY), while the CDC 9-track tape drives can read or write at 800 cpi (HD), or 1600 cpi (NT or PE).

Tape manufacturers often specify the maximum density at which their tapes should be used in terms of flux changes per inch (fci). Because the NRZI recording technique produces, at most, one flux change per character, a tape certified at 800 fci is sufficient for recording at 800 cpi. For the PE recording technique, the fci certification should be twice the cpi density, that is, 3200 fci certification for 1600 cpi usage.

6.8.4 Blocks and Interblock Gaps

A block is the group of frames written by one output instruction issued to the tape control unit. Blocks are sometimes referred to as physical record units or PRUs. Blocks are separated from one another by an interblock gap (or inter-record gap), which is a section of "blank" tape (no flux changes) approximately 3/4 inch long.

Under SCOPE/HUSTLER the minimum block length is 48 bits (8 frames for 7-track tapes, 6 frames for 9-track tapes). Any block shorter than 48 bits is considered noise and is ignored. The maximum block length depends on the procedures used to write the tape. See the discussion of data formats in Section 6.9.

Whatever the length of a block, it is the smallest unit of information that can be physically transferred to or from the tape. Recall that the tape travels at 150 or 200 inches per second and that data is packed at densities of up to 1600 characters per inch. Obviously it is impossible to stop the tape between adjacent frames, so it is impossible to read or write one portion of a block and then another portion of the same block. The interblock gap represents the length of tape necessary to stop and restart the tape motion between operations, typically about 3/4 inch.

Theoretically a 2400-foot tape recorded at 800 cpi can hold more than 23 million characters, but this does not take into account the tape used for interblock gaps. Because of this spacing between blocks, the capacity of a tape depends on the length of the blocks, i.e., the number of frames per block. To illustrate, suppose a tape is written in blocks of 100 characters each. At 800 cpi, the physical length of each block would be 1/8 inch plus 3/4 inch for the interblock gap. Thus, about 85 percent of the tape would consist of interblock gaps and only 15 per cent would consist of data. By increasing the block size to 1280 characters (the standard block size for coded SCOPE tapes), these percentages are almost reversed—70 per cent data and 30 percent interblock gaps.

6.8.5 File Gaps

Blocks are grouped into partitions. Partitions are terminated by a file gap, also called a physical file mark. A file gap is a unique block written six inches after the last block of the file and 3/4 inch before the first block of the next file. The pattern written for a file gap depends upon the recording method as follows:

7-track NRZI: (all densities)
Even-parity block containing one 17_8 character and LRC character (longitudinal redundancy check character)

9-track NRZI: (800 cpi (HD) density)
Odd-parity block containing one 23_8 character, with no cyclic redundancy check character

9-track phase encoded: (1600 bpi (PE) density)
Control block of 82 flux reversals at 3200 reversals/inch in tracks 2, 5, and 8. Tracks 1, 3, 4, 6, 7, and 9 (parity) are DC erased. (DC means direct current. This produces tracks with no flux changes.) On a read operation a file gap is recognized containing 64 to 256 flux reversals on tracks 2, 5, and 8.

Under SCOPE/HUSTLER the file gap serves as an end-of-partition only for the stranger tape formats (see Section 6.9.3). File gaps appear on standard SCOPE tapes only to delimit system-written labels from data blocks.

6.8.6 Beginning-of-Tape and End-of-Tape Markers

Magnetic tapes must have several feet of unused tape at the beginning and end of the reel so that the tape can be threaded onto the tape drive. The beginning and end of the usable portion of tape are marked by aluminum reflective strips glued to the uncoated side of the tape. The tape drive uses a photocell to detect the presence of these strips.

The beginning-of-tape (BOT) marker is placed about 15 feet from the physical beginning of the tape. Since this is the position to which the tape is positioned when it is loaded onto the tape drive, it is sometimes called the load point. Rewinding the tape returns it to the load point if the tape is unlabeled; if the tape is labeled, a rewind positions it immediately after the label. The end-of-tape (EOT) marker is placed about 25 feet from the physical end of the tape. When the tape drive detects this marker, the system will start end-of-volume procedures as soon as the current block has been read or written.

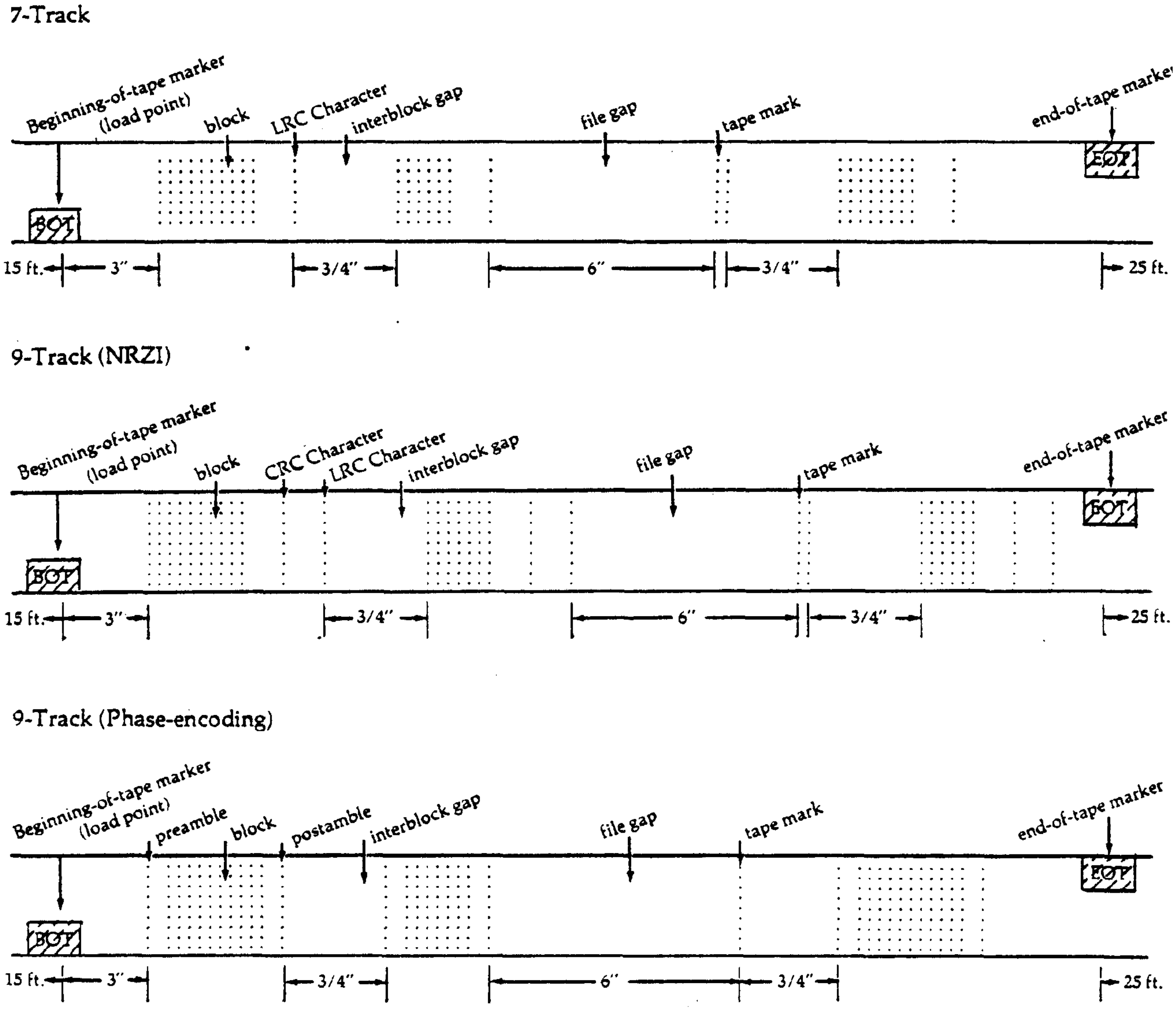
6.8.7 Parity

As noted in Section 6.8.1, a 7-track tape frame consists of six data bits and one parity bit. The parity bit is set by the tape drive control unit in such a way that the number of "1" bits in each frame is always even or always odd. The choice of odd or even parity (corresponding to binary or coded mode) is either implicit in the type of input/output procedure or an explicit user option. For example, the FORTRAN formatted READ and WRITE are even parity (coded) procedures, while the unformatted READ and WRITE are odd parity (binary) procedures. The parity of BUFFER IN and BUFFER OUT statements, on the other hand, is specified by a 1 or a 0 as the second argument. With 9-track tapes, however, the frame parity is always odd, regardless of whether the data is written by coded or binary procedures.

As a tape is read, the tape drive control unit checks the parity of each frame to verify that it agrees with the mode of the request. If it does not, the control unit signals the operating system that a parity error has occurred. See Section 6.14 for a discussion of SCOPE/HUSTLER tape parity error procedures.

Additional checking of the tape data is made by adding one or two check characters to the end of each data block when the NRZI recording technique is used. For 9-track tapes, a "cyclic redundancy check character" (CRC character) is generated as the data is written, and is written four spaces after the last data character in the block. For both 7- and 9-track tapes, a "longitudinal redundancy check character" (LRC character) is generated and written four spaces after the CRC character on 9-track tapes, or four spaces after the last data character on 7-track tapes. The value of the LRC character is calculated such that each track will contain an even number of "1" bits. In other words, the longitudinal parity is always even, whereas the vertical parity for 7-track tapes depends on whether the block is created with a binary or a coded write.

9-track phase-encoded tape blocks do not contain any check characters. They do contain a 41-character preamble and postamble. The preamble consists of 40 characters which are "0" in all tracks, and a single character of a "1" in all tracks. The postamble is a mirror image of the preamble; that is, the first character is a "1" in all tracks and the last 40 characters are "0" in all tracks.


7-Track

9-Track (NRZI)

9-Track (Phase-encoding)

Figure 6-6: Physical Data Structures

6.8.8 Coded Tapes

A coded tape is one on which information is recorded as external BCD (Binary Coded Decimal), EBCDIC (Extended Binary Coded Decimal Interchange Code), or ASCII (American Standard Code for Information Interchange) characters. The 6-bit BCD codes are normally used for 7-track tapes, and the 8-bit EBCDIC and ASCII codes for 9-track tapes. External BCD, EBCDIC, and ASCII are industry standards established to facilitate the exchange of data between computers having different internal coding schemes.

When a user reads a coded 7-track tape, the tape drive control unit converts the data from external to internal BCD, and then the channel converter converts the internal BCD to Display code. The opposite conversions are performed on output. Except for the colon, which is changed to a "0" in the translation, these conversions will be transparent to the user.

To prevent the user from reading a coded 7-track tape in binary mode, or vice versa, SCOPE/HUSTLER adopts the industry standard of writing coded tapes in even parity and binary tapes in odd parity. When a tape is written in even parity, the six data bits of each frame must contain at least one "1" bit to avoid the possibility of a group of all-zero frames, which would appear to the tape drive control unit as an interblock gap. Consequently none of the Display codes convert to an external BCD 00. The following table lists some peculiarities of this conversion from disk to tape and back to disk.

	Display Code	Internal BCD	External BCD	Internal BCD	Display Code
binary zero	00	16	16	16	00
zero (0)	33	00	12	00	33
colon (:)	63	12	12	00	33
end-of-line	0000	1672	1632	1672	0000
blank followed by]	5562	1672	1632	1672	0000

When a user reads a 9-track S or L format tape in coded mode, the tape drive controller converts the data from 8-bit ASCII or EBCDIC to a 6-bit Display code value. The opposite conversion is performed on output, using a 64-character subset of the ASCII or EBCDIC character set. See Appendix A for the conversion table.

Unlike 7-track tapes, 9-track coded tapes are written in odd parity.

6.8.9 Binary Tapes

A binary tape is one on which information is recorded in the same form as it is represented in memory. For 7-track tapes, each central memory word is disassembled into ten tape frames, six bits per frame. With 9-track binary tapes, each pair of 60-bit words is unpacked into 15 frames on the tape. There is no code conversion involved in reading or writing a binary tape.

Because the data bits within a frame or a sequence of frames may be all 0's, binary tapes are recorded in odd parity. This ensures that each frame contains at least one "1" bit. If even parity were used, a group of all-zero frames would be interpreted as an interblock gap.

The motivation for using binary mode tapes is to avoid superfluous conversion between internal and external forms. For example, a binary tape may contain character data stored in 6-, 7-, or 8-bit character codes, numeric data stored in fixed-point or floating-point format, or object code output from a compiler. Because these internal forms are machine-dependent, binary tapes are usually inappropriate for exchanging data with another computer.

6.9 Logical Data Formats

SCOPE/HUSTLER is capable of processing magnetic tapes in any of three data formats: standard SCOPE format, stranger (S) format, and long block stranger (L) format.

SCOPE format is assumed unless the S or L parameter is specified on the REQUEST control statement (see Section 6.5.1). The format selected determines the record structure of the tape; that is, it determines the format of the tape blocks. Each format can accommodate binary or coded data (see Section 6.8.8 and 6.8.9), and each can be processed as labeled or unlabeled (see Section 6.11).

6.9.1 SCOPE Tapes

The record structure of SCOPE tapes is essentially the same as that of disk files except for the block size, which is 128 central memory (CM) words for coded mode tapes and 512 CM words for binary mode tapes. Like disk files, SCOPE tapes are organized into one or more SCOPE sections, where each section consists of a sequence of blocks terminating in a short or zero length block. In other words, all tape blocks are a uniform length (either 128 or 512 CM words), except for the last block of each SCOPE section.

SCOPE records, which typically represent card images and print lines, are packed into 128-word blocks. To produce uniform length blocks, records are split between blocks wherever necessary. A file of records consists of at least one SCOPE section.

On output, the system automatically appends a 48-bit level mark to the last block of each SCOPE section. If the section happens to be an exact multiple of the block size (so that the last block of the section is maximum length), the level mark is written as a separate block, which is said to be "zero length" because it contains no data. On input, the level marks are removed from the data and the level number contained in the mark is passed to the user as part of the end-of-section status information. Thus, the presence of the level marks is transparent to the user, unless the tape is read as a stranger tape or under the control of a non-SCOPE operating system.

The level mark, sometimes referred to as an end-of-section (EOS) mark, contains a number from 00_8 to 17_8 specifying the level of the end-of-section. (The level mark contains the following information: $55233552275400\mathit{ll}_8$ for all 9-track tapes and binary 7-track tapes or $20202020202020\mathit{ll}_8$ for coded 7-track tapes, where *ll* is level of section.) Level numbers may be used to group sets of sections within a hierarchy of up to 16 levels (see Section 4.2.4). The section or set of sections terminated by a level 17 EOS forms a partition. That is, the level 17 EOS is equivalent to an end-of-partition (EOP) mark. The system ensures that an EOP mark is written as a zero length block so that it can be overwritten without loss of data if the file is subsequently extended. File gaps appear on SCOPE tapes only in conjunction with system labels (see Section 6.11.2).

SCOPE-ANSI file trailer labels (EOF1) and SCOPE-ANSI volume trailer labels (EOV1) are used to indicate end-of-information and end-of-volume, respectively, for **unlabeled as well as labeled SCOPE tapes**. The EOF1 label is automatically written whenever a file is closed, rewound, or backspaced following a write operation. The EOV1 label is written whenever a file is continued onto another reel of tape.

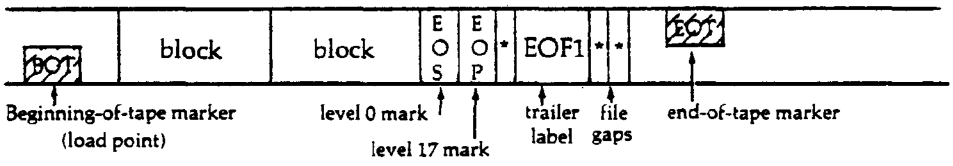


Figure 6-7: Unlabeled SCOPE Tape Containing one Partition

6.9.2 Multifile SCOPE Tapes

A multifile volume set is a tape or a set of tapes containing more than one pair of file header and trailer labels. The purpose of multifile tapes is to allow the user to store a large number of files on a series of tapes without having to remember how many sections or partitions must be skipped in order to position a tape to a specific file.

Because each file trailer label is a SCOPE end-of-information and because the system does not permit the user to read or skip forward beyond end-of-information, special procedures are required to access any file written after the first file of a multifile tape.

Multifile SCOPE tapes cannot be created under SCOPE/HUSTLER. When the file trailer label is written, the tape drive backspaces over it. Any additional write will destroy this label. As a result only one valid file trailer label will remain on the tape.

6.9.3 Stranger Tapes

The stranger tape formats are used to read or write tapes that have been received from, or will be sent to, installations that do not operate under a SCOPE-compatible system. Two stranger tape formats exist; these are S format and L (long block) format. The file structure of S and L tapes relies only on the concepts of tape blocks and standard file gaps. For example, certain COMPASS statements that normally process SCOPE sections or SCOPE records will each read or write a single tape block when the S or L format is declared (see Section 6.16.3). FORTRAN, although normally operating as described for COMPASS, can employ Cyber Record Manager to provide automatic blocking and deblocking of S or L tapes. COBOL provides methods for blocking and deblocking user sections with S and L tape blocks. The effect of various input/output procedures on S and L tapes will be described in more detail below.

S tapes may contain blocks ranging in size (in frames) as shown below.

	7-track	9-track
binary	8-5120	6-3840
coded	8-5120	8-5120[1]

The L format is identical to the S format except that the maximum block size is restricted only by the user program's buffer size. Blocks within a file may vary or they may be fixed at an arbitrary length, depending on the procedures used to create them. Because the peripheral processors and the data channels transfer data in 12-bit bytes, the blocks must be a multiple of two characters for 7-track tapes. Nine-track tapes may have an odd character count.

Partitions are delimited by a standard file gap, which can be written with an end-of-partition operation such as the ENDFILE statement in FORTRAN or the WRITEF macro in COMPASS. On input, a file gap is treated as an end-of-partition equivalent to a SCOPE level 17 end-of-section. All other blocks read under the S or L format are treated as SCOPE sections of level 0.

If an S or L tape is requested as unlabeled, SCOPE/HUSTLER automatically writes four consecutive file gaps following the last block written by the user. These gaps however, are not equivalent to a SCOPE end-of-information. The program which reads the tape must be designed to recognize the file gaps (or a user-defined mark) or it will continue to read data beyond the last valid block. If the tape is labeled, the user is protected by a file trailer label which signals end-of-information, or by a volume trailer label which initiates continuation reel processing.

[1] When a 9-track tape is written in coded mode, the 6-bit Display code character is converted to an 8-bit character. In binary mode, no such conversion is made, four 6-bit characters will fit in three 8-bit frames.

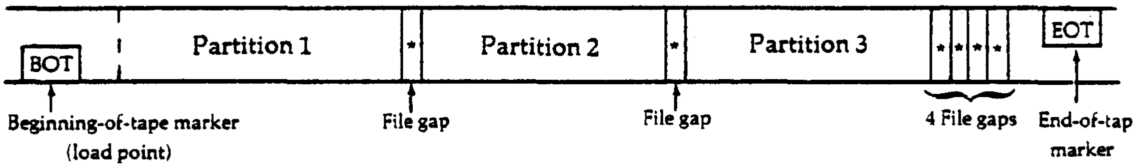


Figure 6-8: Unlabeled Stranger Tape Containing Three Partitions

6.9.4 Specifying Block Sizes for Stranger Tapes

FORTRAN Procedures

The formatted READ statement treats each block of an S or L tape as a record (e.g., a card image). If a block exceeds the maximum of 150 characters, or that defined on the PROGRAM statement, FORTRAN will ignore it and attempt to process the next block.

The formatted WRITE statement writes one S or L tape block and omits the SCOPE end-of-line mark. A single record per block is written. The maximum record size for a formatted WRITE is 137 characters or the record size specified on the program header statement; FORTRAN will append blank characters as necessary to make each record an integer multiple of central memory words. For example, if a FORMAT statement indicates a record of 137 characters, the record will be padded with three trailing blank characters when it is written to the tape.

The BUFFER IN and BUFFER OUT statements also process one S or L tape block, and they are capable of processing binary tapes as well as coded tapes. There is no maximum block size for L tapes; the maximum is 512 central memory words for S tapes. If the S format is requested for a tape containing blocks that are longer than 512 CM words, an attempt to BUFFER IN those blocks causes the following: (1) the block is not transferred to the buffer and the tape is positioned at the start of the next block, (2) the UNIT function returns "READY, NO ERROR" and (3) the LENGTH function returns zero. An attempt to BUFFER OUT a block longer than 512 central memory words on an S tape causes a fatal execution error.

Binary (unformatted) READ and WRITE statements should be used on S and L tapes only in conjunction with Cyber Record Manager and an appropriate FILE statement. The binary READ and WRITE statements normally use an internal blocking scheme which is usable only on SCOPE tapes.

Cyber Record Manager may be used to provide a number of different blocking and tape formatting options in conjunction with standard FORTRAN coded READ and WRITE statements, as well. See Section 6.10 for a brief discussion of Cyber Record Manager.

COBOL Procedures

Records are blocked according to the BLOCK CONTAINS clause of the File Description entry for S and L formatted tapes. Or, if the BLOCK CONTAINS clause is omitted, each user section is written as one block.

The user may read and write blocks that are shorter than the block size, but never longer. The block size may be specified in terms of the number of records or the number of characters it contains. When the block size is stated in terms of records and the records are variable length, the length of the longest record is used to compute the block size. Tape blocks on 7-track tapes must contain an even number of characters. If an odd block size is specified, an extra character will be generated; the actual character will be unpredictable.

On SCOPE tapes, each user record is terminated by an end-of-line mark (Display code 0000 in the lower 12 bits of a central memory word). The end-of-line mark is not used on S or L tapes, so there is no system-supplied delimiter separating the records within a block. Consequently, records are not split between blocks on S or L tapes as they are on SCOPE tapes. For example, if the user specifies a block size of 250 characters and a record size of 100 characters, the tape will be written with 200-character blocks.

6.10 Cyber Record Manager

Cyber Record Manager (CRM) is a CDC utility whose primary tasks are to provide record and block input/output for files. Various types of records, blocks, and file organizations can be identified by CRM, and several methods of record boundary determination and blocking are available.

Following is a brief description of Cyber Record Manager. An example of CRM used to read and write a blocked stranger tape is found in Section 6.15.3. For a complete discussion of Cyber Record Manager, see the CDC *Cyber Record Manager Reference Manual.*

6.10.1 File Organizations

Record Manager supports six file organizations; only one of these, sequential files, is discussea here in relation to magnetic tapes.

Sequential files are tape-like in structure. Records are placed in the order of presentation; physically, a record immediately follows the previous record. Given the location of one record, the location of the next record is determined in relation to the given record only. A sequential file may extend across any number of volumes and may only be accessed sequentially.

6.10.2 Block Types

Four block types exist for use with sequential files. Each block type provides a different method of grouping records within a tape block.

Internal Blocking Type I
Each block contains 5120 characters, with internal control words which describe the records and block. Records can span blocks.

Character Count Block Type C
Each block contains 5120 characters if written in binary mode, or 1280 characters if written in coded mode on a SCOPE tape. Records can span blocks.

Record Count Block Type K
This is the preferred method of blocking stranger tapes. Each block contains an integer number of records per block (specifiable by the user) of a fixed record length (also user-specifiable). Records may not span blocks.

Exact Records Block Type E
Each block contains as many complete records as possible without exceeding a user-specified maximum block size. Records cannot span blocks.

6.10.3 Record Types

Eight record types are defined in Cyber Record Manager. Each record type represents a different method of describing record boundaries (the beginning and end of records). Not all record types may be used with the various block types; refer to the *Cyber Record Manager Reference Manual* for more information.

Decimal Character Count Type D
A field in each record specifies the length of that record.

Fixed Length Type F
This is the preferred record type for stranger tapes. Each record is a fixed length. If necessary, short records are padded with blanks to achieve the desired length.

Record Mark Type R
Each record ends with a user-defined "end-of-record" character.

SCOPE Logical Record Type S
Each record is a SCOPE section (in Cyber Record Manager, called a a logical record). On stranger tapes, this is a single block.

Trailer Count Type T
Each record may contain a number of fixed length trailer portions. The number of these trailer portions is recorded within the record.

Undefined Type U
This record type allows processing of otherwise undefined record types.

Control Word Type W
Each record is prefaced by a control word used to identify record boundaries.

Zero Byte Type Z
Each record is terminated by a 12-bit byte of zeros in the low-order bits of the last word of the record.

6.11 Tape Labels

Tape labels are 80-character blocks that identify the reel of tape and the files it contains. SCOPE/HUSTLER supports ANSI standard labels. Tapes containing labels of any other format are considered unlabeled. (Exception: 3600 labels may be read by SCOPE/HUSTLER. See Section 6.11.2.) To create an unlabeled tape, the user must specify the Z and S parameters on the REQUEST control statement (see Section 6.5.1). To read an unlabeled tape the user must specify the Z parameter on the REQUEST statement. In the absence of a Z parameter, the tape will be processed as a SCOPE-ANSI labeled tape. See Section 6.11.2.

6.11.1 Advantages of Labeled Tapes

Since system-supported labels are automatically written and read by the operating system, their presence is transparent to the user. Nonetheless they provide the user with a number of advantages.

1. Labels enable the system to verify that the correct tape has been mounted and assigned. This phase of label checking is a locally designed feature and is described in detail in Section 6.13.1.

2. Labels improve the reliability of end-of-information and end-of-volume procedures for S and L tapes. Labels are used for this purpose on SCOPE tapes even when they are requested as unlabeled.

3. The system records a block count in the trailer labels and subsequently checks this count against the number of blocks read.

4. The user can specify information to be written into the labels or into additional user labels which may later help to identify the contents of the tape.

The only significant disadvantage of using labeled tapes is that they may be incompatible with the label format of another system. For example, both SCOPE/HUSTLER and the MTS system at the University of Michigan use label formats based on the ANSI standard, but MTS records two sets of file header and trailer labels for each file while SCOPE/HUSTLER records only one. If a tape is to be sent to another installation, it may be best to create an unlabeled tape and record the pertinent identification information separated from the data by a file gap.

6.11.2
SCOPE/ANSI Label Formats

The Computer Laboratory supports SCOPE-ANSI labels which are designed to conform with the ANSI standard submitted by the X3.27 Committee in 1966. Since 1966, the ANSI standard for magnetic tape labels has been revised to include several additional types of labels and several additional fields for existing labels. The SCOPE label format uses four types of labels, each identified by the first four characters of the label.

VOL1	Volume header label
HDR1	File header label
EOF1	File trailer label
EOV1	Volume trailer label

Each label is 80 characters in length and is recorded at the same density as the data. The only local modification to the SCOPE label format is the inclusion of the problem number field in characters 13 through 19 of the volume header label.

The following terms are used to describe the tape file structure defined by SCOPE-ANSI labels.

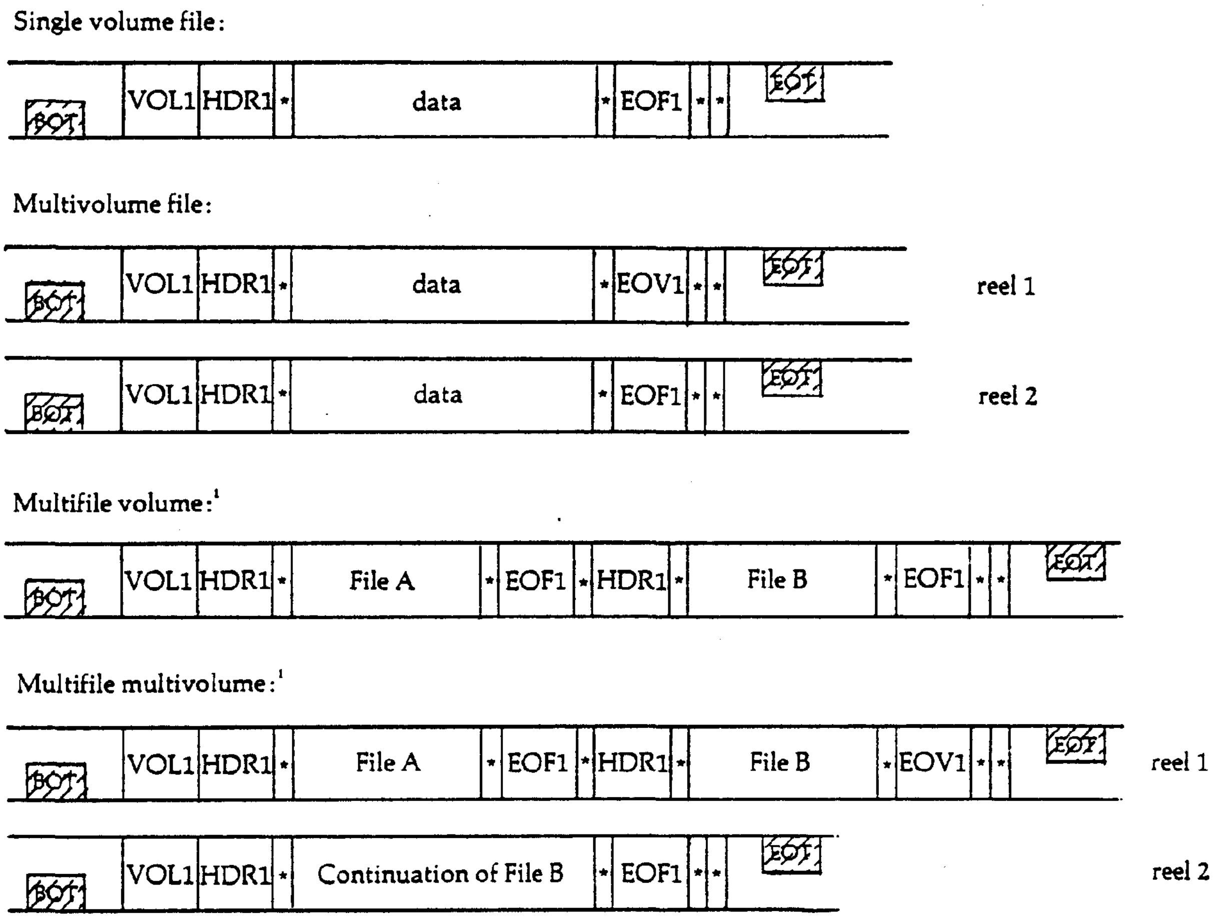
volume — synonymous with a reel of magnetic tape.

file — in the sense used here, the information delimited by a file header label and a file trailer label. This unit of information may in turn consist of one or more partitions, each terminated by a level 17 end-of-partition for SCOPE tapes or a file gap for S and L tapes. The file trailer label functions as a SCOPE end-of-information mark.

volume set — the tape reel or sequence of reels on which a file or set of files is recorded. If a volume set is recorded on more than one reel, it is called a multireel volume set. If a volume set contains more than one pair of file header and trailer labels, it is called a multifile volume set. Since multifile volume sets contain more than one end-of-information, they require special handling in order to position the tape to a particular file. See Section 6.9.2 for a discussion of multifile tapes.

file gap the same special block used to delimit files in the S and L data formats. File gaps separate label blocks from data blocks. They are considered part of the labels and are read and written automatically by the system. An EOV1 label followed by a double file gap indicates end-of-volume in a multi-volume set; an EOF1 label followed by a double file gap indicates the end of the set.

The structure of SCOPE labeled tapes is shown in Figure 6-9. The label identifier is used to denote the entire 80-character record, and asterisks are used to denote file gaps.


Single volume file:

Multivolume file:

Multifile volume:[1]

Multifile multivolume:[1]

Figure 6-9: Structure of SCOPE Labeled Tapes

The exact format of the information recorded in each label is described in Appendix I, but the following will summarize their function and content.

Volume header label:

The VOL1 label is the first block of each reel. The '1' in the label identifier has no relation to the sequence of reels in a multireel volume set; the reel number is contained in the file header label. The volume header label contains the visual reel name (VRN), the user's problem number, and the density of the data.

[1] Multifile tapes are not supported at MSU and their use is discouraged.

File header label:

The HDR1 label follows the volume header label and, for a multifile volume set, precedes each subsequent file as well. If a file spans two or more reels, the header label for that file is repeated after the VOL1 label for each continuation reel. A file header label contains the following fields: the file name, the multifile name, the reel number for multireel files, the file position number for multifile volume sets, the edition number, the creation date, and the expiration date. Normally many of these fields will be blank. Section 6.11.3 explains how to specify the information written into these fields.

File trailer label:

The EOF1 label terminates the file defined by the preceding HDR1 label; it is the SCOPE end-of-information for the file. The file trailer label contains the same information as the preceding header label plus a count of the blocks contained in the file.

Volume trailer label:

The EOV1 label is written when a file is continued onto another reel of tape; that is, it is written whenever the end-of-tape (EOT) reflector is sensed before the file trailer label is written. The format of this label is identical to that of the file trailer label except for the label identifier.

3600 Labels (obsolete)

Tape labels created on the CDC 3600, though now obsolete, can be read by SCOPE/HUSTLER. The structure of the 3600 label is described in Appendix I.

To read a tape with a 3600 label, specify the 'Y' parameter on the REQUEST control statement.

6.11.3 Label Processing

For most users, the use of labeled tapes requires no special considerations. After requesting the tape, FORTRAN or COMPASS programmers can read or write data as they would for any sequential disk file. A COBOL programmer uses the LABEL RECORD IS clause to specify whether the tape has standard system labels, user data-name labels, or no labels. Tape labels created without any extra consideration will contain blank characters in many of the informational fields, but will still supply the security and reliability advantages listed in Section 6.13.

The information recorded in a label is taken from several sources: the REQUEST statement parameters, the job PNC, the File Environment Table (FET, described in Section 4.3.1), and values calculated by the system. Under SCOPE/HUSTLER, most label information is merely for user documentation. When a labeled tape is read, any user information in the label fields of the File Environment Table is checked against the corresponding fields of the file header label. If there is a discrepancy, the operator will be instructed to type "DROP" or "GO" to either abort the job or to continue processing. At MSU the operator types "GO," since initial label checking (performed when the tape is assigned) ensures that the correct tape is mounted and that it has the VRN specified on the user's REQUEST statement. If the user wishes to abort the job in such a situation, he/she must inform the operator or perform his/her own label checking.

Normally when a labeled tape is created, label fields are left blank, or are given default values as shown in Appendix I. The FORTRAN or COBOL user can insert values into the label fields. The LABEL subroutine call in FORTRAN or the LABEL RECORD IS clause in COBOL may be used.

FORTRAN

The FORTRAN subroutine LABEL is called with

CALL LABEL (u,fwa)

u the unit number of the file or the file name in left-justified zero-filled Display code.

fwa the first element of a four-word array that contains label information in the format described below.

59	47	29	23	17	0
File Label Name (first 10 characters)					Word 1
File Label Name (last 7 characters)				Position Number	Word 2
Edition No.	Retention Cycle	Creation Date			Word 3
Multifile Name			Reel Number		Word 4

File Label Name 17 alphanumeric characters (starting with a letter), left-justified with zero fill, identifying the file. If this field is zero when labels are written, the tape labels will contain blanks in the file label name field. The file label name is always checked when the tape is read.

Position Number 3 numeric digits specifying the sequence of the file in a multifile set. Like the multifile name, this field may be ignored.

Edition 2 numeric characters identifying successive editions of the same file. If zero when labels are written, 01 is assumed. If zero when labels are checked, the edition number field is ignored.

Retention Cycle 3 numeric characters specifying the number of days that the tape is to be protected from accidental destruction. This value is added to the creation date to compute the expiration date when labels are written. The expiration date is checked when the tape is opened for writing. The default value is zero. A value of 999 is considered an infinite retention period.

Creation Date 5 numeric characters: the first two specifying the year, the other three specifying the Julian day of the year (001 to 366). If omitted or zero when labels are written, the current date is used. If zero when labels are checked, the creation date is ignored.

Multifile Name 6 alphanumeric characters (starting with a letter), left-justified with zero fill. Since the multifile capability is not implemented in SCOPE/HUSTLER, this field should be ignored.

Reel Number 4 numeric characters specifying the sequence of reels in a multireel file. If omitted when labels are written, reel number 0001 is assumed. The reel number is incremented by one at the conclusion of volume trailer label processing for each reel. It is reset to 0001 when the file is closed. If omitted when labels are checked, the reel number is ignored.

Example:

```
PROGRAM SAMPLE (TAPE1,TAPE2)
DIMENSION INFO(4)
DATA INFO /10LSURVEY 2 R, 7LAW DATA, 5L01999, 0/
CALL LABEL (2,INFO(1))
.
.
.
WRITE (2,201) X
```

The call to LABEL specifies the file label name as SURVEY 2 RAW DATA, the edition number as 01, and the retention cycle as 999. The system inserts default values for the creation date and the reel number.

Note: This subroutine will fail if called under MSU Record Manager; it will work correctly with Cyber Record Manager.

COBOL

For standard labeled tapes the following LABEL RECORD clause is used.

```
       { RECORD IS   }
LABEL  { RECORDS ARE }   STANDARD

      [                [ { ID             }        { literal-1     } ]
      [ VALUE OF       [ { IDENTIFICATION }   IS   { data-name-1   } ]

                       [                           { literal-2     } ]
                       [   DATE-WRITTEN       IS   { data-name-2   } ]

                       [                           { literal-3     } ]
                       [   EDITION-NUMBER     IS   { data-name-3   } ]

                       [                           { literal-4     } ]
                       [   REEL-NUMBER        IS   { data-name-4   } ]

                       [                           { literal-5     } ] ]
                       [   RETENTION-CYCLE    IS   { data-name-5   } ] ]
```

The notation used here is that of the *COBOL Reference Manual.*

The label fields have the same meanings as those described above for FORTRAN. Note that the ID field (the file label name) is required but all other label fields are optional. Default values are supplied by SCOPE/HUSTLER for omitted fields, as described above.

Example:

```
IDENTIFICATION DIVISION.
          .
          .
          .
DATA DIVISION.
FILE SECTION.
FD     FILE-1
       LABEL RECORDS ARE STANDARD
       VALUE OF ID IS "SURVEY 2 RAW DATA"
       EDITION IS "02"
       RETENTION IS "999"
```

The OPEN and CLOSE Functions

In order to understand when labels are created and checked, the user must be acquainted with the open and close functions. The open function prepares a file for processing by setting certain fields of the File Environment Table (see Section 4.3.2), setting up a File Name Table entry if one does not already exist, reading in the index for random-access files, and processing labels on magnetic tape files. The close function is used to write an end-of-information mark, write out the index for a random file, and rewind or return the file.

6.11.4 Label Creation

A volume header label and a file header label are written when a file is opened for writing as the first operation on the tape. An open for writing operation will read labels if a read operation has been done or if a previous open to write operation has been done. A file trailer label is written whenever the tape is closed and the file is positioned after a newly written record.

To the FORTRAN programmer, this means that volume and file header labels are written if the first action on a file is a write. A file trailer label will be written when there is reverse motion on the tape following a write.

6.11.5 Label Checking

Any tape containing recognizable labels will be checked when the tape is assigned to make sure that the label VRN matches the requested VRN and, if a write-enable ring was also requested, that the label PN matches the job PN.

The file header label is read and its contents are checked against the specified tape label fields of the File Environment Table a tape file that is currently closed is opened to be read. To the FORTRAN programmer, this means that label checking takes place when the first action on a file is a read or a skip-forward, or if the file was rewound just prior to the read. The user is reminded, however, that the system checking of the file header label has no real effect, since the operator will instruct the system to continue program execution in spite of label discrepancies.

Note: Labels are processed unless the tape is requested with nonstandard labels (NS parameter). If a labeled tape is requested unlabeled (using the Z parameter), the tape will be positioned past the header labels. If an unlabeled tape is requested as labeled and the first action is a read or a skip, the system will look for a label, discover it missing, and post the "TYPE DROP OR GO" message to the operator. Normally, the operator will respond "GO."

Caution: If an unlabeled tape is to remain unlabeled, it should always be requested using the Z parameter on the REQUEST statement. Otherwise, a write operation at the beginning-of-tape marker will cause the tape to be labeled.

6.11.6 Printing the Label: PRINTLB Control Statement

The PRINTLB card is used to print out the contents of tape header labels.

```
PRINTLB,lfn.
```

lfn the local file name of the tape.

In order to use PRINTLB successfully, the tape must be requested with the S, Z, and NS parameters on the REQUEST control statement. PRINTLB reads the first blocks and, if it recognizes a SCOPE-ANSI label, copies the information to OUTPUT.

Example:

```
PNC
id,MT1.
PW=password
REQUEST,TAPE,VRN=861,S,NS,Z.
PRINTLB,TAPE.
6/7/8/9
```

6.12 End-of-Volume Procedures

The end-of-volume procedures performed by SCOPE/HUSTLER depend on whether:

1. the tape is a SCOPE or a stranger tape,
2. the tape is labeled or unlabeled,
3. the user processing option has been requested by a COBOL program (see Section 6.12.1) or COMPASS program (see Section 6.16.5).

Normally—which is to say, when the user processing option is not requested—SCOPE/HUSTLER processes continuation reels automatically. When the end-of-volume is detected, it will rewind and unload the tape and request the next reel of tape specified in the VRN list of the REQUEST statement. If a continuation reel is not specified, or if the VRN list is exhausted, the system will abort the job and issue an appropriate dayfile message. The assignment of a continuation reel, like any other tape request, is subject to the tape security checks described in Section 6.13. After the tape is assigned, SCOPE/HUSTLER will also perform the standard SCOPE label procedures for tapes requested as labeled. Then it will resume the operation that was in progress on the preceding reel.

During a write, the end-of-volume condition is signaled by the EOT reflective marker. The following table summarizes the steps performed when the EOT marker is sensed during the last block written.

	Labeled SCOPE	Unlabeled SCOPE	Labeled Stranger	Unlabeled Stranger
Write EOV trailer label	x	x	x	
Write four file gaps				x
Rewind and unload reel	x	x	x	x
Request next reel	x	x	x	x
Write header label(s)	x		x	
Continue function	x	x	x	x

During a read, the end-of-volume condition is signaled by the EOV or EOT trailer label if the tape is labeled or SCOPE formatted. If the tape is an unlabeled stranger tape, the end-of-volume condition is signaled by the EOT marker. Detection of the end-of-volume is not as reliable on unlabeled stranger tapes; see Section 6.12.2.

6.12.1
User Processing Options

The user may request alternative end-of-volume procedures in COBOL by selecting the user processing option. This is done when a USE procedure is declared for checking and/or preparing tape labels (generally for user data-name labels). See the discussion of the USE statement in the *COBOL Reference Manual.*

6.12.2
End-of-Volume Errors on Unlabeled Stranger Tapes

When reading an unlabeled stranger (S or L) tape, a spurious end-of-partition status may be returned at end-of-volume. This is an unavoidable problem caused by the following sequence of events.

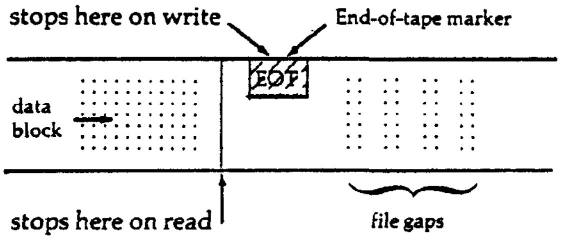


Figure 6-10: End-of-Volume Error on Unlabeled Stranger Tape

When the tape is written, the data section may end before the EOT marker, but the tape may coast far enough to detect the marker and initiate end-of-volume procedures. These procedures, as described earlier, consist of writing four file gaps (end-of-partition marks) and requesting a continuation reel.

When the tape is subsequently read, the tape may not coast quite as far and so not detect the EOT marker. The next read would detect both the EOT marker (end-of-volume) and the first file gap (end-of-partition). Note that the effect is random. On another run, the tape may coast farther, detect the EOT marker when the last data section is read, and initiate reel swapping without an end-of-partition indication.

The only way to avoid this situation is to use either labeled or SCOPE formatted tapes.

6.13
Tape Security System

The SCOPE/HUSTLER tape security system, designed locally and implemented in 1970, provides the following types of protection. The first item applies to all tapes, but the second and third items apply only to labeled tapes (i.e., tapes written with either SCOPE-ANSI or MSU 3600-style labels).

1. It ensures that the user cannot write on a tape unless he/she has explicitly requested a write-enable ring.

2. It ensures that the operator mounts and assigns the correct reel, i.e., the VRN that was requested.

3. It ensures that the user cannot write on a tape unless the user's problem number matches that of the label on the tape, or no problem number is specified in the label.

The tape security system cannot prevent any user from **reading** most tapes. Nor can it prevent a user from mistakenly overwriting one of his/her own tapes as long as the operator mounts the reel as specified.

A "File Protect" ring can be installed on a reel of tape that physically prevents the operator from inserting a write-enable ring. There is a nominal charge for this service; a special tool is required to install or remove the "File Protect" ring. Inquire at the Service Window in Room 208 Computer Center.

6.13.1 Security Procedures

The mechanics of the tape security system are relatively straightforward. When the operator attempts to assign a magnetic tape drive to the user's job, the system checks for the presence or absence of a write-enable ring. If the tape was not mounted as requested, it is immediately unloaded and the operator is instructed to remove or insert the ring. If the tape is labeled, the system checks the header label to see if it contains the correct VRN. If, in addition, a ring was requested, the system also checks the label to see if it contains the correct problem number. A discrepancy in either case causes the assignment to be rejected. If the discrepancy resulted from an operator error, the operator is allowed to mount the correct reel and reissue the assignment command; otherwise the job must be aborted. This procedure is defined more precisely in the following algorithm.

1. Does the tape have a recognizable label? (regardless of whether or not it was requested as labeled)

   If NO, go to 5.

2. Does the label VRN match the requested VRN? If not, is the label VRN blank or zero? Or was the REQUEST VRN omitted (i.e., will a scratch tape be assigned; see Caution on scratch tapes in Section 6.5.1)? A YES to any of these questions constitutes a "correct VRN."

   If NO, go to 7.

3. Was a ring requested?

   If NO, go to 5.

4. Does the label PN match the job PN? If not, is the label PN blank or zero?

   If NO, go to 6.

5. Was the tape mounted with or without a write-enable ring, as specified by the REQUEST statement?

   If YES, go to 8.

6. Was a scratch tape requested? This test is made when the user is requesting to write on a labeled tape, where the label PN does not match the job PN. It may be that the scratch tape mounted by the operator was labeled by another user. But if the user did not request a scratch tape and the correct VRN was assigned (step 2), then the job is aborted because the user is requesting permission to write on a tape that he/she is not authorized to write on.

   If YES, go to 8.

7. Assignment Error. The tape is unloaded and the operator is informed of the error. If the operator was responsible for the error, he/she will correct the condition and reassign the tape drive. Otherwise the job will be dropped after an informative dayfile message has been entered from the operator's console.

8. Tape is assigned.

6.13.2
Changing the Label PN

Unless the REQUEST statement Z or NEWPN=0 parameter is specified, tapes created under SCOPE/HUSTLER are written with labels containing the user's problem number (PN). Once the labels are written, only jobs submitted with the same PN are allowed to request the tape with a write-enable ring. Note that when creating a labeled tape, the user can specify a PN other than the job PN through use of the REQUEST statement NEWPN parameter. Specifying NEWPN=0 will cause the PN field of the labels to be left blank and thus allow any user to request the tape with a write-enable ring.

If you later wish to change the label PN, note that the NEWPN option is effective only if (1) the tape is being rewritten from the beginning, and (2) the job to rewrite the tape is submitted with the PN currently recorded in the labels.

If you do not want to change the other contents of the tape you should first make a copy of the complete tape on a second, or backup, tape. Only when the copy is found to be complete and correct should the PN in the label of the original tape be changed. Sample jobs to change the PN of a binary 7-track SCOPE tape containing five partitions are shown below.

```
PNC (for problem number 010001)
id,MT2.
REQUEST,TAPE1,VRN=872.
REQUEST,TAPE2,VRN=873,NEWPN=010002,RW.
COPYBF,TAPE1,TAPE2,5.
REWIND,TAPE1,TAPE2.
COMPARE,TAPE1,TAPE2,999,17.
6/7/8/9
```

The COMPARE control statement verifies that a complete duplicate of the tape has been successfully made. If the compare shows no errors, the following job can be run.

```
PNC (for problem number 010001)
id,MT2.
REQUEST,TAPE1,VRN=873.
REQUEST,TAPE2,VRN=872,NEWPN=010002,RW.
COPYBF,TAPE1,TAPE2,5.
REWIND,TAPE1,TAPE2.
COMPARE,TAPE1,TAPE2,999,17.
6/7/8/9
```

If you do not have access to the PN used to create the labeled tape, you must request the Computer Laboratory to "blank-label" the tape (i.e., rewrite the label records with blanks).

Note: **Blank-labeling a tape effectively destroys the contents of the tape. You should be sure to have a backup copy of the data before requesting the blank-labeling operation.**

You may request that a tape be blank-labeled by filling out a Tape Service Request form available at the Service Window in Room 208. All tape service requests must specify the PN and the account number associated with the tape when it was submitted for storage; your department can supply this information. After the tape has been blank-labeled, you should rewrite it with a non-blank label if you want to restrict write access. Otherwise any user will be able to request the tape with a write-enable ring.

6.14
Parity Error Procedures

The tape drive control unit constantly checks for errors during all read and write operations. It does this by checking the number of "1" bits in each frame to see if it agrees with the odd or even parity specified by the user's read/write request. It also keeps a count of the "1" bits in each track to verify the even-parity longitudinal redundancy check character (LRC character) at the end of each block. A discrepancy in either test constitutes a **parity error**.

Parity errors detected during a write are usually caused by damaged or dirty tape, although the tape drive or accumulations of oxide on the read/write heads may also be at fault. Read parity errors, in contrast, are sometimes the fault of the user rather than the tape or tape drive. Two types of user errors are possible:

1. Reading the tape in the wrong mode (binary instead of coded or vice versa). Although this type of parity error is unrecoverable, it will be diagnosed. The message "MT xx RD ERR READ OPP MODE" will appear in the dayfile for 7-track tapes.

   On 9-track tapes, when a binary SCOPE tape is read in coded mode, the message "DEVICE CAPACITY EXCEEDED" will appear; when a coded 9-track SCOPE tape is read in binary mode the user will receive the message, "RECORD FRAGMENT." The "DEVICE CAPACITY EXCEEDED" message also appears if an S tape with blocks of 3841 to 5120 characters is read in binary instead of coded mode.

2. Requesting the wrong density. Because the density used to read labeled tapes is taken from the volume header label, this type of error can occur only on 7-track tapes requested as unlabeled. Nine-track tape drives automatically select correct density.

The tape drive is designed to check each block as it is being read or written. If an error is detected within a block, SCOPE/HUSTLER immediately initiates a sequence of procedures to reread or rewrite that block unless "NR" is specified on the REQUEST statement. If the block cannot be reread or rewritten without error after many retries, the error is declared to be "unrecoverable."

The remainder of this section is divided into three parts. The first two detail the write and read error recovery procedures. The third explains how unrecoverable errors are handled.

6.14.1
Write Parity Error Recovery

The SCOPE/HUSTLER peripheral processor routines that handle magnetic tape input/output provide extensive parity error recovery procedures, which include noise bracketing of unusable tape. SCOPE/HUSTLER defines a noise record to be any tape block of less than eight characters. Noise bracketing consists of writing special, "system noise records" (SNRs) at the beginning and end of an unwriteable length of tape. When the tape is read, the system ignores all information between SNRs. This applies only to 7-track tapes; 9-track tapes do not use noise bracketing.

Noise bracketing greatly improves the user's ability to read and write tapes in spite of errors in the tapes and/or tape drive hardware. Tapes containing noise brackets, however, may not be read properly under other operating systems. For this reason, the dayfile message "MTxx WARNING—BRKT WRITTEN-NON STANDARD TAPE" is issued whenever noise brackets are written. In addition to MSU's SCOPE/HUSTLER system, tapes containing noise brackets can be read normally at any CDC 6000, 7000, or CYBER series installation operating under SCOPE 3.3, SCOPE 3.4, NOS/BE, or 7000 SCOPE 2.0. If a tape is to be shipped to an installation other than these, the NB (no bracketing) parameter should be included on the REQUEST control statement when the tape is created. The NB parameter selects an alternate method of write parity error recovery.

For those interested, here is the algorithm for write parity error recovery:

1. Perform a "controlled backspace" (reverse erase) function. This erases the bad block and should leave the tape positioned to the interblock gap following the last good block.

2. Perform a reverse read followed by a forward read to ensure that the controlled backspace went far enough, or that it did not go too far and erase part of the last good block.

3. If the block read in Step 2 was a nonstandard noise record (not an SNR), backspace over it and repeat Step 2.

4. The tape is now positioned at the end of the last good block.

5. Perform a "skip bad spot" function, which erases about six inches of tape in the forward direction, and then rewrite the block. If there are no parity errors in the rewrite, stop and issue the "WRITE ERROR RECOVERED" message. Otherwise backspace to the position arrived at in Step 4.

6. If the NB parameter was specified for the file, go to Step 12.

7. Write a system noise record (SNR) consisting of four Display code zeros (converted to External BCD for coded tapes), and set the SNR counter to zero.

8. Perform a "skip bad spot."

9. Increment the SNR counter by one, convert it to a four-digit octal Display code number, and write a system noise record consisting of these four coded digits (converted to external BCD for coded tapes).

10. If there were parity errors during the writing of the system noise record, erase it and go to Step 8.

11. Attempt to write the data. If successful, stop and issue the "WRITE ERROR RECOVERED" message. The write error recovery procedure is finished. Otherwise, backspace to the position arrived at in step 4.

12. Perform repeated "skip bad spot" functions and write attempts until the block is successfully written or until 25 feet of tape have been erased (about 50 rewrite attempts). If the record is successfully written the "WRITE ERROR RECOVERED" message is written in the dayfile. Otherwise unrecoverable parity error procedures are initiated (see Section 6.14.3).

In any of the above steps, a block is considered to have been successfully recovered if the block and the SNR (if any) preceding it can be read in both forward and reverse directions.

6.14.2 Read Parity Error Recovery

Here is the algorithm for read parity error recovery. If the block is read without error during Steps 1, 3, or 4, the system issues an informative dayfile message and continues reading normally.

1. Backspace to the beginning of the block and reread it in the specified mode (i.e., binary or coded). If necessary, try this step twice.

2. Backspace to the beginning of the block and reread it in the opposite mode. If successful, transfer the data to the user's buffer and return the unrecovered parity error status code (FET error code 04), along with the dayfile message "MT xx RD ERR READ OPP MODE." Repeat this step one more time if the parity error persists. Note that the data transferred to the buffer will not be properly converted. If the user attempts to read a coded tape in binary mode, the characters will be received in internal BCD. If the user attempts to read a binary tape in coded mode, the data will be interpreted as external BCD characters and translated to Display code.

3. Try Step 1 two more times.

4. Backspace over the three blocks preceding the bad block and then skip forward three blocks and read the fourth. This procedure, known as "on-the-fly" recovery, is an attempt to clean the tape of any obstructions that may be causing the parity error. This step is tried up to eight times.

5. Stop—unrecoverable parity error.

When 9-track NRZI tapes are read, the cyclic redundancy check character (CRC) is also checked. If it is incorrect, it will indicate which track is in error. If more than one track is indicated, the error is unrecoverable. The block is reread if a single-track error, and any frame with vertical parity error is corrected in the track indicated as bad. If this generates a good CRC character compare, the data is given to the user as good.

On 9-track phase-encoded tapes, if a bit is dropped in one track of a frame, it is automatically regenerated from the data in the other 8 bits. If more than one track drops a bit, parity error recovery procedures are used.

Whenever the system encounters a noise record (a block of less than eight characters) on a 7-track tape that it does not recognize as a system noise record (SNR), it checks to be sure it is not a misread data block. First, the block is reread in both the forward and reverse directions, and then one of the following actions is taken:

1. If the block appears as noise when read in both directions, search forward to the next non-noise block. The dayfile message "MT xx NOISEWARNING n" is issued for each non-standard noise record encountered.

2. If the block is found to be a file mark, begin normal end-of-file processing.

3. If the block can be reread as a non-noise block, the block is recovered and the dayfile message "MT xx RD RVD NOISE IN IRG" is issued.

4. If the block can be reread as a non-noise block but parity errors occur, switch to the normal read parity error recovery algorithm listed above.

5. If the block appears as noise in a forward direction and non-noise in the reverse direction, attempt to read the tape in reverse and then reverse the data to proper order as it is transferred to the user's buffer.

6.14.3
Unrecoverable Parity Errors

The handling of unrecovered read and write parity errors depends on whether error processing has been requested for that file. This is requested automatically for input files used in FORTRAN programs and is a user option for files used in COBOL (see below) and COMPASS programs (see Section 6.16.4). The error processing is not in effect when the file is used by a COPY utility or most of the other system routines.

If the error processing option is not set and an unrecoverable parity error is encountered on the file, the system suspends execution of the job and displays a message at the operator's console. If the operator types "RECHECK," the system will repeat the error recovery algorithm. If he/she types "GO," the system will ignore the error and resume normal execution. If several unrecoverable errors occur in succession, the operator will type "DROP" to abort the job.

Several unrecovered parity errors occurring on a tape may indicate that either the tape itself or a tape drive is defective. If all efforts to read a tape have been tried without success, the Computer Laboratory will, on a case-by-case basis, adjust a tape drive in an attempt to accommodate the tape. Users should contact the shift supervisor at the Service Window in Room 208.

FORTRAN Parity Error Procedures

The FORTRAN input routines set a flag when an unrecovered read parity error occurs on a tape-resident file. If this flag is not checked before the next read request for the file, the program will abort with an appropriate error message. When using formatted or unformatted READ statements, the IOCHEC function should be used to check the error status of the previous read. The UNIT function should be used after BUFFER IN statements. Write parity errors are not checked by FORTRAN; instead, the decision to stop or continue job execution is left to the operator, as described above.

The block containing the unrecovered read parity error will be transferred to the user's program, but in the case of BUFFER IN, the LENGTH function will return zero. The data in this block may or may not be usable; it may happen that only a parity bit is bad and all the data bits are good. If the data is bad, though, the parity error may lead to other errors, such as error number 78: "ILLEGAL DATA IN FIELD."

Another note of caution is that, for SCOPE tapes, the FORTRAN routines called by the formatted and unformatted READ statements usually read several blocks ahead of the one being processed by the user's program. For example, if IOCHEC returns non-zero (parity error) after the user's program has executed a formatted READ statement five times, the error is more likely to be the 250th block than in the 5th. The exact position of the error cannot be readily determined by the user's program. FORTRAN does not read ahead for stranger tapes or for BUFFER IN statements. Cyber Record Manager can be used to suppress the read ahead process.

COBOL Parity Error Procedures

In COBOL, the error processing option is set by specifying a USE AFTER STANDARD ERROR PROCEDURE statement in the Declaratives Section of the Procedure Division. If this declarative is not in effect for the offending file, parity error processing will be handled by SCOPE/HUSTLER in the fashion described above. Otherwise, parity error processing is controlled by the COBOL input/output routines as follows:

1. For stranger tapes, the user may specify (following the USE AFTER STANDARD ERROR PROCEDURE statement) routines to be executed in the event of a read parity error; if a file with the implement-name ERRFILE is defined, the bad block will be placed in the ERRFILE block area, along with the implementor-name of the file from which it was read.

2. For SCOPE tapes and write parity errors, COBOL will terminate the run.

6.15 Special Problems

The following sections deal with problems for which users frequently seek help from the consultants.

6.15.1 Changing the Number of Tape Tracks

Physically, there is no difference between 7- and 9-track tapes. Either type of tape can be mountea on either type of tape drive if the tape has been certified for 3200 fci (see Section 6.8.3). However, if a 9-track tape is read on a 7-track drive, or vice versa, gibberish and unrecoverable parity errors are returned. Since labels would not be recognized, the tape security system would be voided. To prevent this, 9-track tapes have distinctive paper labels glued to the tape reel. The operators will not mount any tape so labeled on a 7-track drive, neither will they mount any tape without such a label on a 9-track drive. This process is subject to human error, and is one of the reasons that users should maintain backup copies of important data.

A user who wishes to use a 7-track tape as a 9-track tape, or vice versa, must make such a request at the Service Window in Room 208. Since a blank-label operation (see Section 6.13.2) is performed when this is done, it is important to note that any data on the tape is effectively destroyed, and, if desired, a backup copy of the data should be made beforehand.

6.15.2 Exchanging Tapes with Other Sites

Compactness and portability make magnetic tape a convenient medium for transporting information in a computer-readable format. The exchange of magnetic tapes between two sites is complicated, however, by the wide variety of schemes used to structure the data. This section identifies common sources of incompatibility and suggests ways of avoiding them. It also suggests how to handle some typical problems.

Hardware Compatibility

The CDC 7-track tape drives are compatible with IBM 727 and 729 I through VI tape drives, and they can read and write at densities of 200, 556, or 800 cpi. The CDC 9-track drives can read and write at densities of 800 or 1600 cpi. Users who wish more detailed information about the tape drives should contact the consultants, who will arrange an appointment with one of the Control Data engineers.

Documentation Suggestions

Insufficient information causes most of the problems users experience when they try to read a tape received from another site. The following items should be provided. Do not hesitate to request this information when you send for a tape.

1. Number of tracks. MSU can handle 7- and 9-track tapes.

2. Density. MSU can handle 200, 556, or 800 cpi for 7-track, 800 or 1600 for 9-track. If there are header blocks (labels), are they recorded at the same density as the data? SCOPE-ANSI labels are always written (and read) at the same density as the data.

3. Odd or even parity. Again, if there are special header blocks, are they the same parity as the data? SCOPE-ANSI labels are always even-parity on 7-track tapes regardless of the parity of the data blocks.

4. Character codes. Is the data encoded in BCD, EBCDIC, ASCII, or CDC Display code? Seven-track coded (even-parity) tapes created by SCOPE/HUSTLER are written in 6-bit external BCD codes. Nine-track coded tapes are written in ASCII or EBCDIC. On input, the system assumes that even-parity tapes contain external BCD codes and converts them to Display code. Binary (odd-parity) tapes are written in whatever codes the user's program uses internally. The internal coding scheme of tapes written at another site will rarely be compatible.

5. Labels. Are there special header blocks other than standard SCOPE-ANSI labels? If so, provide a complete description of their contents, format, and location. Include their density, parity, and use of file gap delimiters.

6. Continuation reels. If data spans two or more reels, what is the structure of the data around the EOT reflective marker and at the beginning of the next reel?

7. Record Structure.

   a. Block length. Are the blocks (physical records) fixed or variable-length? If variable, what is the maximum and minimum lenth? How can the length of a particular block be determined?

   b. Record length. Are the records fixed or variable-length? If variable, what is the minimum and maximum, and how can the length of a particular record be determined?

   c. Blocking scheme. Is each block a record or are two or more records contained in each block? Can a record span two blocks?

   d. Delimiters. How are end-of-partition, end-of-information, and end-of-tape indicated? Are there non-data characters within the data? For example, coded SCOPE tapes contain special bytes indicating end-of-line, end-of-section, and end-of-partition.

8. Contents. Describe the number of files and their position on the tape. If a file contains data rather than programs, describe the fields within each record. Is each field numeric or alphabetic?

Format Suggestions

The following suggestions are offered on the premise that it is usually much easier for the sender to create a format compatible with the receiver's computer than for the receiver to decipher an incompatible format.

If you send for a tape from a non-CDC site, request that it be written in even-parity using the external BCD character set for a 7-track tape, or odd parity using EBCDIC or ASCII character set for 9-track; these are industry-wide standards and should be available. If a 7-track tape arrives written in odd-parity or using a different coding scheme, you must write a program to convert the data to the CDC Display code used internally by SCOPE/HUSTLER. When you write a tape using a coded (even-parity) operation, such as the formatted WRITE statement in FORTRAN, the data is automatically converted from Display code to external BCD for 7-track tapes, or EBCDIC or ASCII for 9-track tapes, by the tape drive hardware. To write a tape using other codes, you can generate the codes with your program and use a binary (odd-parity) mode operation to transfer the data to tape without conversion.

Header and trailer labels, although useful to the system that writes them, are generally more trouble than they are worth to other systems. Similarly, minor differences in end-of-volume procedures can cause major difficulties. If more than one reel is needed, split the data into two files rather than span the file across two reels. In general, when you send a tape to a site other than a CDC 6000 or 7000 installation operating under a version of SCOPE, write the tape as an unlabeled stranger (S or L) tape and suppress the noise bracketing feature for unrecovered write parity errors. In other words, include the following options on the REQUEST control statement for 7-track tapes:

```
REQUEST,lfn,VRN=vrn,RW,S,Z,NB.
```

where RW means "read-write," S means "stranger tape format," Z means "unlabeled," and NB means "no bracketing" for write parity error recovery. The density will be 556 cpi.

For 9-track tapes, the REQUEST statement would be similar:

```
REQUEST,lfn,VRN=vrn,RW,S,Z,NB,EB.
```

where EB provides EBCDIC characters; use AS instead if ASCII characters are desired. (The AS or EB parameter indicates that the tape is 9-track.) The density will be 800 cpi.

Although most systems are flexible enough to handle a variety of record and block structures for magnetic tape, the simplest, and therefore most universal, format consists of fixed-length, unblocked records. Unless the tape can be written in SCOPE format, this will also be the easiest to read on the 6500. "Unblocked" means that each block is a single record. This is the format produced by FORTRAN formatted WRITE statements and BUFFER OUT statements when the REQUEST statement S or L parameter is specified; that is, each statement writes one block. "Fixed-length" means that the records are all the same length. To accomplish this, the FORTRAN programmer can adjust the FORMAT statements or the buffer limits of the BUFFER OUT statements to pad shorter records with trailing blanks.

Blocked formats (two or more records per block) can be created or read by using Cyber Record Manager. COBOL programmers have the BLOCK CONTAINS clause to produce blocked S or L tapes automatically. Before creating a blocked format, be sure you understand the blocking conventions used at the other site.

Line images are automatically blocked into 1280-character blocks for coded SCOPE tapes, but sending a SCOPE tape to a non-SCOPE installation will only cause them needless problems.

If possible, records and blocks should be a multiple of the memory word-size of the computer that will read the tape. For instance, for tapes sent to another CDC 6500 site, records should be a multiple of 10 characters.

Incompatible Labels

When you send for a magnetic tape, ask the sender to omit the special header blocks known as labels. Although the labeling scheme followed by SCOPE/HUSTLER was designed to conform with the ANSI standard, the standard has since changed to include additional types of labels and additional fields within the labels. The standard also provides fields for local options, which are a common source of conflict. Here are two typical problems caused by incompatible label formats.

1. Wrong VRN. Columns 5-10 of the VOL1 label contain the visual reel name (also called the volume serial number) by which the tape is identified to the operator. SCOPE/HUSTLER checks the contents of this field against the REQUEST statement VRN parameter whether or not the tape is requested as labeled. Consequently, if a tape arrives with 1234 in the VRN field of the volume header label, it must be requested with 'REQUEST,lfn,VRN=1234.' etc. Temporary tapes are normally stored with a physical identification of TMPnnn. Therefore, special arrangements must be made with the operator so that he/she will mount the correct reel when 1234 is requested. This may be done by preceding the REQUEST statement with a statement of the form 'HAL,PAUSE,USE TMP152 FOR 1234.'

2. Wrong PN. The contents of columns 13-19 of the standard VOL1 label are not normally used, but under SCOPE/HUSTLER they contain the user's problem number. If a tape with a recognizable volume header label contains some other information in these columns, you will be unable to request the tape using the RW parameter. To gain write permission, you must have the tape "blank-labeled" by the operator. See Section 6.13.2.

Incompatible Codes

Unless your tape is written in even-parity and is coded in external BCD on 7-track, or ASCII or EBCDIC on 9-track, or was written in binary by a CDC 6000 SCOPE system, you will have to write a program to convert the data to CDC Display code.

In FORTRAN, the general method of these conversions is to construct an array of Display code characters, such that subscripting the array with a character from the source set yields the corresponding Display code. For example, the letter 'A' is represented as 01 (octal) in Display code and 61 (octal) in external BCD. To translate external BCD to Display code, you would build an array—say ICHAR—such that ICHAR(61B) contains the integer 01B. To set up the conversion, you would first read a record into an array—say IN—and then unpack each 6, 7, or 8-bit character from IN and store it as a separate element of another array—call it IBYTE—in R format (right-justified with zero fill). The conversion can then be stated as IBYTE(I)=ICHAR(IBYTE(I)).

The simplest case of code conversion is one where the data on a 7-track tape is written in odd-parity external BCD or some other 6-bit code. You would then read the tape in binary mode, and the data would be transferred to memory exactly as it appears on the tape.

When data is written in even-parity, the conversion program is complicated by the fact that the system automatically treats the data as external BCD and converts it to Display code. So, to construct the ICHAR array, you would have to map each character from the source code to external BCD to Display code, and then from the mistranslated Display code to the proper Display code.

Codes constructed of 7 or 8 bits present two problems. First, the unpacking algorithm is more complex because some of the characters will be split between two words. Secondly, 7- and 8-bit character sets are two or four times as large as 6-bit character sets. Thus, you must decide how to handle the characters that are not defined in Display code.

6.15.3 Reading and Writing Blocked Stranger Tapes

Magnetic tapes exchanged between computer sites are often written in a blocked format; that is, each block may contain more than one record. The advantage of blocked tapes is that the data occupies less space, or rather, the larger block size means there is less space taken up by interblock gaps. The disadvantage of blocked stranger tapes (S or L tapes) is that additional control statements are required to unblock them, except through COBOL.

Blocked stranger tapes typically contain fixed-length records. The FORTRAN procedure for reading or writing these is straightforward and will be demonstrated shortly. If blocked records are variable-length, each block will normally contain some control bytes indicating the length of the block and/or each record within the block. For example, below is a diagram of the IBM "VB" (variable length, blocked) tape structure,

BDW	RDW	record 1	RDW	record 2	RDW	record 3 . . .

Figure 6-11: IBM Tape Structure

where BDW is a 4-character Block Descriptor Word containing the length of the block, and RDW is a 4-character Record Descriptor Word containing the length of the record.

As an illustration, suppose a tape arrives with fixed-length 80-character card images blocked 100 cards per block. The FORTRAN procedure for unblocking the tape is simple: Cyber Record Manager is used to describe the blocked file and to do the actual unblocking and the FORTRAN program need only read in and write out each line in 8A10 format. A sample program and job deck are shown below. Note that because the 8000-character blocks exceed the 5120-character maximum allowed for S tapes, the L format must be declared. In this example, the user writes the unblocked card images on a SCOPE tape; he/she could just as well write them on a disk file and catalog it as a permanent file or dispose it to punch.

```
PNC
id,MT2.
REQUEST,TAPE1,VRN=TMP123,L,Z,HY.
REQUEST,TAPE2,VRN=5061,RW.
FTN.
FILE,TAPE1,BT=K,RT=F,RB=100,FL=80,MBL=8000.
LDSET,LIB=FORTRAN/CRM.
LDSET,FILES=TAPE1.
LGO.
7/8/9
          PROGRAM UNBLOCK (TAPE1,TAPE2,OUTPUT)
          DIMENSION LINE(8)
    1     READ(1,2) LINE
    2     FORMAT (8A10)
          IF(EOF(1).NE.0)GOTO 3
          WRITE(2,2) LINE
          GO TO 1
    3     CONTINUE
          END
6/7/8/9
```

A job to write a stranger tape with the same format of the above example is similar. The FORTRAN program would be the same; only the control statements identifying which tape is read from or written to are changed:

```
PNC
id,MT2.
REQUEST,TAPE1,VRN=5061.
REQUEST,TAPE2,VRN=TMP123,L,Z,HY,RW.
FTN.
FILE,TAPE2,BT=K,RT=F,RB=100,FL=80,MBL=8000.
LDSET,LIB=FORTRAN/CRM.
LDSET,FILES=TAPE2.
LGO.
7/8/9

     FORTRAN program as shown above

6/7/8/9
```

6.15.4 Copying Random Files to Magnetic Tape

The most commonly used input/output procedures (such as the READ, WRITE,. and BUFFER statements in FORTRAN) are designed to read and write file sections in sequential order. However, SCOPE/HUSTLER provides procedures for processing "random" or "indexed" files. These terms refer to the way a file is processed rather than to characteristics of file structure. An indexed file can be read without regard to positioning; instead the user identifies the section to be read by a key (a name or number), and SCOPE/HUSTLER automatically positions the file to that section.

Whether a file (either on tape or disk) is processed by random-access or sequential access methods is determined by a flag, the RP (random processing) flag. In most cases this flag is not set and the file is processed sequentially as would any tape-resident file.

If the RP flag is set when a file is created, the file is thereafter considered a random file. Random files created by this method are called "SCOPE random files." If the RP flag is not initially set, SCOPE/HUSTLER flags the file as sequential, but provides random-access processing if the RP flag is subsequently set. Random files created in this way are called "indexed sequential," or "pseudo-random" files.

Programs that use index manipulation facilities almost invariably create SCOPE random files. An example is the UPDATE program library. Programs that perform their own index manipulation (e.g. EDITOR) normally use "pseudo-random" files.

Given the distinction between these two types of random files, we can state that problems in copying random files to and from magnetic tape typically arise only when copying SCOPE random files. There are two causes for these problems.

1. SCOPE/HUSTLER copy utilities always generate a sequential output file; that is, the RP flag is not set when the file is created.

2. COPYCF and COPYBF write an end-of-partition mark at the end of the output even when none is present on the input file. SCOPE random files terminate in a single level 0 end-of-section; if an additional partition mark is appended by COPYBF, SCOPE/HUSTLER will assume the index to be missing the next time it opens the file.

The HAL utility FCOPY can be used to circumvent these problems. Designed specifically for fast real-time copying, FCOPY reads and writes simultaneously. It also copies a file exactly (appending no extra section marks), and it will, upon your request, create a random output file. The following example shows how a random UPDATE program library can be stored to and retrieved a tape.

To copy to tape:

```
ATTACH,OLDPL,RANDOMPL.
REQUEST,TAPE,VRN=123,RW.
HAL,FCOPY,I=OLDPL,O=TAPE.
```

To copy to disk:

```
REQUEST,TAPE,VRN=123.
HAL,FCOPY,I=TAPE,O=OLDPL/RND.
CATALOG,OLDPL,RANDOMPL.
UPDATE.
```

Note that the '/RND' suffix indicates that the output file is to be a SCOPE random file.

A similar situation exists when the user wishes to copy word addressable files (described in the *Cyber Record Manager Reference Manual*, Chapter 2) to and from magnetic tape. FCOPY has a 'WA' suffix that is used for such files in the same way as the 'RND' suffix discussed above.

6.15.5
Recovering Data From Blank-Labeled Tapes

As stated in Section 6.13.2, blank-labeling a tape effectively destroys the contents of the tape. A labeled tape has the following structure:

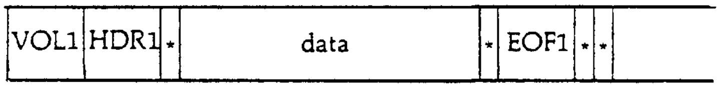


Figure 6-12: Labeled Tape

where

VOL1 is a volume header label
HDR1 is a file header label
EOF1 is a file trailer label
* is a file gap

Blank-labeling consists of writing a header label sequence followed immediately by a trailer label sequence, thereby indicating an empty tape, as follows:

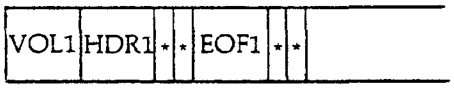


Figure 6-13: Empty Tape

The term "blank" refers to the fact that most of the information normally recorded in the labels is left blank.

Blank-labeling writes a trailer label over what was previously data, so that approximately the first twenty inches of data (8000-15000 characters) are irretrievably lost. The rest of the data remains intact, but efforts to recover it are impeded by the trailer label, which acts as an end-of-information indicator. The simplest method of getting past this label is to request the tape as an unlabeled stranger tape (using the Z and S options on the REQUEST statement). In this mode, the labels are treated as data and the file gaps as end-of-partitions.

Skip the first four partitions and the first section of the next partition. Then copy the remainder of the fifth partition to another stranger tape. Since end-of-partitions do not look like file gaps, they will not be affected by the copy.

Example:

PNC	
id,MT2.	
REQUEST,A,VRN=100,S,Z.	Requests blank-labeled tape as stranger, unlabeled.
REQUEST,B,VRN=101,S,RW.	Requests scratch tape as labeled, stranger.
SKIPF,A,4,17.	Skip four file gaps to end of first trailer label sequence.
SKIPF,A,1.	Drop the first block of data, because it is probably incomplete and would cause a record fragment error when read as a SCOPE tape.
COPYCF,A,B.	Copy through next file gap to recover remaining data.
REWIND,B.	
HAL,FILEDMP,I=B,BCD=O,R=5.	Examine first five blocks to determine how much data was lost.
6/7/8/9	

Recovering data from blank-labeled tapes of other formats—such as binary files or indexed files—would require other, or additional, steps.

6.16
COMPASS Procedures

The following sections discuss COMPASS procedures for several magnetic tape operations. They have been removed from the body of the chapter to facilitate the transfer of this information to another reference manual at a later time.

6.16.1
MSUREQ Macro

Tape requests using the REQUEST control statement are discussed in Section 6.5.1.

COMPASS programmers may issue object-time tape requests. The majority of the tape attributes are specified in the FET of the file, although the tape VRN(s) is stored in the file's buffer. The actual request is issued using the MSUREQ macro. Although the standard SCOPE macro, REQUEST, has been retained for compatibility, its use is discouraged. The REQUEST macro does not allow the user to specify the ring status, the NEWPN parameter, or more than one VRN. The format of the MSUREQ macro call is

```
MSUREQ          lfn[,rng][,NB]
```

lfn the local file name, i.e., the address of a FET.

rng RO for read-only (no ring)
RW for read-write (ring)
AUTO for automatic assignment (disk requests only)
The default is RO.

NB No bracketing—This option selects alternate write parity error recovery procedures in place of the noise bracketing of unusable tape (applies only to 7-track tapes). See Section 6.14.1.

MSUREQ assembles into a call to CPC:

<table>
<tr><td>59</td><td></td><td></td><td></td><td>29</td><td></td><td>17</td><td>0</td></tr>
<tr><td colspan="4">SA1 lfn</td><td colspan="2">RJ</td><td colspan="2">CPC</td></tr>
<tr><td>REQ</td><td>0</td><td>1</td><td colspan="2"></td><td>n+m</td><td colspan="2">000500</td></tr>
</table>

where n	= 0	for RO	and m	= 0	for NB omitted
	= 1	for AUTO		$=10_8$	for NB selected
	= 4	for RW			

Except for the NB and RO/RW/AUTO options, all parameters for the MSUREQ function are held in the FET indicated by lfn, and its buffer. These options can be inserted in the FET at execution time, or may be specified when the FET is created with the appropriate FILEx FET creation macro (i.e., FILEB, FILEC, RFILEB or RFILEC). The following fields of the FET are used.

Word 1: bits 59-18 local file name
bits 17-0 function code (set by CPC)

Word 2: bits 59-48 device type
bits 35-24 disposition code
bits 17-0 address of VRN list (FIRST—pointer to first word of buffer)

The device type field has two parts. To request a magnetic tape, bits 59-54 of the device type field must contain the tape identifier value: 40_8 for a 7-track tape request, or 41_8 for a 9-track tape request. Bits 53-48 of this field specify the format identifier value, which contains the sum of the values for the recording density, the label format, and the data format as follows:

binary		**octal**
bits 53-48		
	7-TRACK TAPES	
xxxx00	HI density, 556 bpi	0
xxxx01	LO density, 200 bpi	1
xxxx10	HY density, 800 bpi	2
xx00xx	unlabeled	0
xx01xx	SCOPE-ANSI labels	4
xx10xx	3600 labels	10
00xxxx	SCOPE data format	0
10xxxx	S data format	40
11xxxx	L data format	60
	9-TRACK TAPES	
xxxx10	HD density, 800 bpi	2
xxxx11	PE density, 1600 bpi	3
xx00xx	unlabeled	0
xx01xx	SCOPE-ANSI labels	4
00xxxx	SCOPE data format	0
10xxxx	S data format	40
11xxxx	L data format	60

The format identifier is calculated by summing the octal values for the density, label, and data format.

To form the device type field it is necessary to concatenate the tape identifier and format identifier values. For example, to request a 7-track unlabeled S tape at 800 cpi, a device type of 4002_8 must be specified. To request a 9-track labeled SCOPE tape at 800 cpi, the device type is 4106_8.

In the FILEx macro, the parameter DTY= is used to specify the device type. For example, DTY=4107 specifies a labeled SCOPE 9-track tape recorded at 1600 bpi. A device type of 4040_8 requests an unlabeled stranger 7-track tape recorded at 556 bpi.

Bits 35-30 of the disposition code field specify the NS, NR and NB options as well as the conversion mode for 9-track tapes. The format is as follows:

binary		**octal**
xxxx01	EB, EBCDIC conversion	1
xxxx10	AS, AS conversion	2
xxx1xx	NB, no noise brackets	4
x1xxxx	NR, no error recovery	20
1xxxxx	NS, nonstandard labels	40

The disposition code is specified with the DSC= parameter on the FILEx macro. For example, DSC=4200_8 specifies a tape recorded in ASCII and expects nonstandard labels which will be processed by the user.

The NEWPN parameter and the VRN list are specified in the file buffer, starting at the location pointed to by FIRST. The format of this parameter vector is:

Word 0 NEWPN the 6 or 7 character problem number (stored in L format) to be written into the tape header label. If this word is binary zero, the PN is taken from the job PNC. To specify NEWPN=0 (no PN), this word should contain 7L0000000.

Word 1 VRN_1

. .

. .

. .

Word k VRN_k The VRN(s) of the tape(s) to be requested. These should be right-justified within the upper six characters of the word. If a VRN is less than six characters, add leading Display code zeros (e.g., 6L000321).

Word k+1 0 the list terminator: a word of binary zero.

Examples:

1. Request tape 1025 as unlabeled, HI density, 7-track, SCOPE formatted, and with a write-enable ring.

```
TAPE1    FILEC      BUF1,513,DTY=4000B,(WSA=WSA1,20B)
BUF1     DATA       0
         DATA       6L001025
         DATA       0
         BSS        510
WSA1     BSS        20B
         .
         .
         .
         MSUREQ     TAPE1,RW
```

2. Request tape 601, with continuation reels 602-603, as 7-track SCOPE-ANSI labeled, SCOPE formatted, and read-only.

```
TAPE2    FILEB          BUF2,1029,DTY=4004B
BUF2     DATA           0
         DATA           6L000601,6L000602,6L000603
         BSSZ           1025
         .
         .
         .
         MSUREQ         TAPE2      default is read-only
```

3. Request tape 102 as an unlabeled stranger 9-track, recorded in ASCII at 1600 bpi.

```
TAPE3    FILEC          BUF3,137,DTY=4143B,DSC=0200B
BUF3     DATA           0,6L000102,0
         BSS            133
         .
         .
         .
         MSUREQ         TAPE3,NB,RW
```

4. Request tape 3889, with continuation reel 3912, as a HY density, labeled SCOPE tape. Request the reels to be mounted with write-enable rings and specify NEWPN=0 to allow write access for any PN.

```
TAPE4    FILEB          BUF4,1029,DTY=4006B
BUF4     DATA           7L0000000
         DATA           6L003889,6L003912
         BSSZ           1025
         .
         .
         .
         MSUREQ         TAPE4,RW
```

6.16.2
COMPASS Label Processing

Label processing is discussed in Section 6.11.3. This section discusses the COMPASS LABEL macro, which allows the user to insert values into the FET fields that describe the label. Words 10-13 of the FET supply the following data:

FET Tape Label Fields

<table>
<tr><th>59</th><th>47</th><th>29</th><th>23</th><th>17</th><th>0</th></tr>
<tr><td colspan="6">File Label Name (first 10 characters)</td></tr>
<tr><td colspan="4">File Label Name (last 7 characters)</td><td colspan="2">Position Number</td></tr>
<tr><td>Edition No.</td><td>Retention Cycle</td><td colspan="4">Creation Date</td></tr>
<tr><td colspan="3">Multifile Name</td><td colspan="3">Reel Number</td></tr>
</table>

File Label Name — 17 alphanumeric characters (starting with a letter), left-justified with zero fill, identifying the file. If this field is zero when labels are written, the tape labels will contain blanks in the file label name field. When the tape is read, the file label name is checked even if the FET field is zero.

Edition — 2 numeric characters identifying successive editions of the same file. If zero when labels are written, 01 is assumed. If zero when labels are checked, the edition number field is ignored.

Retention Cycle — 3 numeric characters specifying the number of days that the tape is to be protected from accidental destruction. This value is added to the creation date to compute the expiration date when labels are written. The expiration date is checked when the tape is opened for writing. The default value is zero. A value of 999 is considered an infinite retention period.

Creation Date — 5 numeric characters: the first two specifying the year, the other three specifying the Julian day of the year (001 to 366). If omitted or zero when labels are written, the current date is used. If zero when labels are checked, the creation date is ignored.

Reel Number — 4 numeric characters specifying the sequence of reels in a multireel file. If omitted when labels are written, 0001 is stored in the FET and written in the tape labels. The FET field is incremented by one at the conclusion of volume trailer label processing for each reel. It is reset to 0001 when the file is closed. If omitted when labels are checked, the reel number is ignored.

Multifile Name — 6 alphanumeric characters (starting with a letter), left-justified with zero fill. Since the multifile capability is not implemented in SCOPE/HUSTLER, this field should be ignored.

Position Number — 3 numeric digits specifying the sequence of the file in a multifile set. Like the multifile name, this field may be ignored.

The tape label fields of the FET may be specified using the LABEL macro. This macro call must immediately follow a FILEB or FILEC macro call containing the LBL parameter. The format of the LABEL macro is

```
lfn      LABEL   fln,ed,ret,create,reel,mfn,pos
```

lfn	local file name (identical to the location field of the preceding FILEB or FILEC macro).
fln	file label name
ed	edition number
ret	retention cycle
reel	reel number
mfn	multifile name
pos	position number

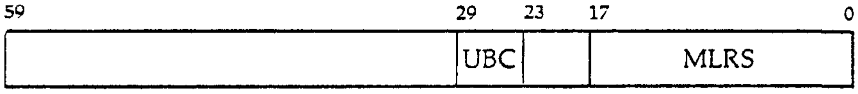
Example:

```
          IDENT           SAMPLE
          ENTRY           START
TAPE2     FILEB           OBUF,1029,LBL
TAPE2     LABEL           (SURVEY 2 RAW DATA),02,999
          .
          .
          .
```

6.16.3 Processing Stranger Tapes in COMPASS

Stranger tapes are discussed in Section 6.9.3. This section describes COMPASS input/output procedures for stranger tapes.

A 7-word FET is mandatory for processing S and L tapes. Word 7 of the FET specifies the unused bit count (UBC) and the maximum logical record size (MLRS) as follows:



The unused bit count allows the user to process records which are not an integer multiple of central memory words. For a READ or READSKP request, SCOPE/HUSTLER will store into the UBC field the number of low-order unused bits in the last data word of the record (pointed to by IN-1). For a WRITE, WRITER, or WRITEF request, SCOPE/HUSTLER will read the UBC field and adjust the length of the record accordingly. For example, to write a record of 124 characters, the user would set IN and OUT to reflect 13 words of data, set the UBC field to 36, and then issue a WRITE or WRITER request. On 7-track tapes, data is transferred in 12-bit bytes, and UBC should contain a multiple of 12; if it is not a multiple of 12 for a write request, SCOPE/HUSTLER will reduce the value to the nearest multiple of 12, but without changing the FET. For 9-track tapes, UBC may be a multiple of 6 (coded mode) or 8 (binary mode).

The MLRS field specifies the maximum number of words that a record may contain. If the data from OUT to IN-1 exceeds MLRS on a write request, the device capacity exceeded code (10B) is returned to the FET, and nothing is written. If the tape contains a block longer than MLRS on a read request, error code 10B is returned and nothing is transferred to the circular buffer. If the MLRS field contains zero for an S tape FET, the maximum value of 512 words is assumed. For L tapes, the default is LIMIT-FIRST-1 for standard reads and writes, and LIMIT-FIRST-2 for READN and WRITEN.

The effects of various input/output requests are outlined below.

READ — Each request transfers one block to the circular buffer. If the buffer does not have at least MLRS words available, the request is ignored. If the block exceeds MLRS words, no data is transferred to the buffer, the tape is positioned to the next block, and error code 10B is returned to the FET. Users who are doing read requests without auto-recall should note that the EOR bit is set after each request. The EOF bit is set after a file gap is sensed. An EOR level 00_8 is returned for each block read on an S or L tape; a file gap on an S or L tape generates an EOR level 17_8 (EOF).

READSKP — For S and L tapes, the READSKP request is identical to the READ request except that the buffer does not need to have MLRS available words. If the buffer is filled before the entire block is transferred to it, the remainder of the record is discarded and the tape is positioned to the next block.

WRITE
WRITER
WRITEF

Each request writes one block, the size of which is determined by the IN and OUT pointers and the UBC field. If the record size exceeds MLRS words or constitutes a noise record (less than 8 characters for 7-track, or 6 characters for 9-track), no data is transferred and error code 10B is returned. A WRITER request is identical to a WRITE request with a level number less than 17_8; that is, the level number is ignored. A WRITEF request or a WRITER request of level 17_8 writes one block (if the buffer is not empty) followed by a file gap.

READN
WRITEN

These procedures, used exclusively for S and L tapes, allow several records to be transferred with a single request. A READN request will transfer as many blocks as possible until end-of-partition or end-of-information, or until the buffer does not contain room for the next record (MLRS+1 words). Reading may continue at tape speed (i.e., without releasing and reloading the PP between records) as long as the available buffer space does not drop below 2(MLRS+1)+1 words. At the beginning of each record, the system inserts a header word stating the length of the record and the number of unused bits in the last data word.

The same header word must be supplied by the user for each record written with a WRITEN request. The header words are not transferred to the tape but merely inform the system of where one record ends and the next begins. Output will be continuous as long as the user stays one full record ahead of the system output routine.

SKIPF
SKIPB

If level 17_8 is specified, the tape is skipped until a tape mark has been read. If any other level is specified, it is assumed to be level 0. Each S or L tape block is treated as a section of level 0.

6.16.4 COMPASS Parity Error Procedures

Parity errors are described in Section 6.14. The handling of unrecovered read and write parity errors depends on whether the error processing (EP) bit is set in the FET of the file. In COMPASS, this processing is optional.

The COMPASS programmer may set the EP bit (bit 44 of the second FET word) by specifying the EPR parameter of the FILEC or FILEB macro. If the EP bit is set, the system will store error code 04 in bits 9-13 of the FET code and status field and return control directly to the program rather than request operator action.

In the case of an unrecoverable read parity error, the bad block is transferred to the buffer and the IN pointer is updated as usual. In the case of an unrecoverable write error, the bad block remains written on the tape but the OUT pointer is not updated.

If file action requests are made through CPC or CIO=, the user may establish an "owncode" routine to be executed when an error condition is detected on the previous request. The owncode option is specified with the (OWN=eoi,err) parameter of the FET creation macro, where eoi is the address of an end-of-information routine and err is the address of the error processing routine. To omit the eoi address, use the form (OWN=,err).

When CPC or CIO= is called for a file action request, the code and status field of the FET is checked to determine if there was an error on the previous request. If so, word 9 of the FET is examined for an error processing address. If word 9 is zero, the error code is ignored and the current request is processed; otherwise, control is transferred to the indicated code.

More precisely, the owncode routine is called by copying the exit word from CPC or CIO= (the one that returns control to the user) into the first word of the routine, putting the contents of the first FET word into X1, and branching to the second word of the routine. By branching to its first word, the user's owncode routine will cause a branch to the point in the program to which control would have returned had the error not occurred (i.e., the instruction word following the call to CPC or CIO=).

The following program illustrates how to set up an error processing owncode routine. The routine in this case does not do very much. It checks the code and status field and, depending on whether or not there was a parity error, either dumps the FET and buffer of TAPE1 or aborts the run.

```
          IDENT       EXAMPLE
          ENTRY       START,TAPE1

          EXT         ERR
TAPE1     FILEB       BUF1,1029,DTY=4004B,(OWN=,ERR),EPR
BUF1      DATA        0,6L003812,0
          BSS         1026

START     MSUREQ      TAPE1,RO
          OPEN        TAPE1,READ,RECALL
          READ        TAPE1,RECALL
          SA1         TAPE1+2
          .
          .
          .
          IDENT       ERR
          ENTRY       ERR
ERR       BSSZ        1                   entry/exit word
          AX1         9                   position error code
          MX2         55                  form mask
          BX2         -X2*X1              extract error code
          SX3         4                   4=parity error code
          IX3         X2-X3               test for 04 code
          ZR          X3,PE               branch if parity error
          ABORT                           otherwise abort

PE        DMP         =XTAPE1,=XSTART     dump buffer and FET
          JP          ERR                 return (would return to SA1
                                          TAPE1+2)
          END
```

6.16.5 COMPASS End-of-Volume Procedures

End-of-volume procedures are described in Section 6.12. The user may request alternative end-of-volume procedures by setting the user-processing (UP) bit in the FET.

In COMPASS, the UP bit can be set by specifying the UPR parameter of the FILEB or FILEC macro. If the UP bit is set, the system will not swap reels when the end-of-reel is detected, but will store the end-of-reel code (02_8) in the FET and exit to the user. CPC will call the end-of-information "owncode" routine if one was specified.

7

SCOPE/HUSTLER Control Statements

Chapter 7 serves as a reference guide to all the control statements that access products in the SCOPE/HUSTLER system having "full support" or "partial support." "Support" refers to the amount of maintenance, improvements and consulting help on the products performed by Computer Laboratory staff. See the *Facilities and Policies Handbook*, Section 7.6, for a full discussion of support.

Included in this chapter are both standard SCOPE statements developed by CDC, and those statements modified at or unique to MSU. Some statements are useful only in batch computing; others may have different functions in batch and interactive modes. This chapter emphasizes batch use, and contains reference to the *Interactive System User's Guide* for users who wish to learn more about the interactive applications of specific control statements. Those statements unique to the interactive system are not mentioned here; they are documented in the *Interactive System User's Guide*, Chapter 2.

Control statements with similar functions, such as language processing or file manipulation, are arranged alphabetically under general categories. Some statements fit appropriately into more than one category; a chapter table of contents is included here to help users find individual descriptions.

The description of each control statement contains the calling sequence, definition of parameters, and special cases, if any. In some cases, control statements are explained in detail in another Computer Laboratory publication or Control Data publication; the user is referred to other sources where appropriate.

For quick reference, turn to Appendix J of this volume, which contains a one-sentence summary of each control statement, as well as detailed notation of the syntax of each. Control statement descriptions can also be obtained from the system by using the HELP utility, as described in Section 7.17.

7.1
Control Statement Processing

This section describes control statements that allow the user to choose alternate methods of control statement processing.

7.1.1
CCEXEC

The CCEXEC control statement allows control statements to be read from a user-created file. This statement is an alternative to the more commonly used statement, EXEC, described in Section 7.1.2. The format is:

```
HAL,CCEXEC,execlfn.
```

execlfn   the name of the local file that contains the control statements to be executed. This is called an "exec" file.

The file execlfn is rewound, and the first section is copied to the file ZZZZEXC. Then the control statements following the CCEXEC statement are copied to the file ZZZZEXC and ZZZZEXC is executed.

CCEXEC files used by interactive jobs may contain both interactive commands and EDITOR directives. EDITOR directives must be preceded by the command 'EDITOR.' and followed by the command 'END.' Interactive commands must be preceded by the command 'MISTIC.' See the *Interactive System User's Guide*, Section 9.1.

Example:

Using the same exec file created in Example 1 of Section 7.1.2, the following job illustrates the difference between EXEC and CCEXEC. Suppose a user wishes to obtain a printed copy of one of the files used in the exec file for this run only. The following job can be run:

```
PNC
job card
PW=password
ATTACH,X,MYEXECFILE.
HAL,CCEXEC,X.
COPYSBF,F,OUTPUT.
7/8/9
data
6/7/8/9
```

The COPYSBF control statement will be executed after the statements on the exec file have completed execution. If the EXEC statement had been used, the COPYSBF statement would not have been executed (see Section 7.1.2).

7.1.2 EXEC

The EXEC statement enables the user to execute control statements from a file other than INPUT, i.e., from a source other than the control section of the job deck.

```
EXEC,execlfn.
```

execlfn the name of a coded local file containing card images of SCOPE/HUSTLER control statements; this file is called an "exec file."

When the EXEC statement is encountered, subsequent control statements are read from the file execlfn. The control statement processor always starts from the beginning of the file, regardless of its current position, but does not reposition the file.

Caution: The EXEC utility does not return to the original control section after executing the control statements from file execlfn. To continue processing the remaining control statements of the job deck, the user can use the HAL utility CCEXEC (see Section 7.1.1) or COMPASS macro EXECM (see Section 8.5.15), rather than the EXEC statement.

Exec files used by interactive jobs may contain both interactive commands and EDITOR directives. EDITOR directives must be preceded by the command 'EDITOR.' and followed by the command 'END.' Interactive commands must be preceded by the command 'MISTIC.' and followed by 'END.' See the *Interactive System User's Guide*, Section 9.1.

Example 1: Implementing an exec file.

The EXEC statement is useful when a sequence of control statements is executed repeatedly. To illustrate, suppose data is frequently merged into a sorted file named CUMULATIVEDATA, which is then processed to generate several tables of summaries. Both the sorted file and the summaries are then cataloged. The following job catalogs a file of control statements that will later be used as an exec file.

```
PNC
job card
PW=password
COPYCR,INPUT,X.
CATALOG,X,MYEXECFILE,RP=999.
7/8/9
ATTACH,A,SORTDIRECTIVES.        file with SORT/MERGE directives.
ATTACH,B,CUMULATIVEDATA.        old data file.
ATTACH,C,SUMMARIES.
ATTACH,D,BINARYPROGRAM.
SORTMRG,I=A.
D.                              Program D writes summary file E and data file F.
SKIPF,C,1.
COPYCF,E,C.
EXTEND,C.
PURGE,B.
CATALOG,F,CUMULATIVEDATA.
6/7/8/9
```

After cataloging MYEXECFILE, the user need only submit the following job each time data are added to CUMULATIVEDATA.

```
PNC
job card
PW=password
ATTACH,X,MYEXECFILE.
EXEC,X.
7/8/9
data
6/7/8/9
```

Auto-Execution

A PN Manager can establish an exec file or program that will automatically execute whenever a user under that problem number logs in or runs a batch job. Execution of the exec file or program can be made optional or mandatory.

The exec file to be used at the start of a job is called the initialization file; it can be created and changed only by the PN manager. The AUTHORF utility is the means for cataloging the initialization file and for controlling its use. This process is described in Section 2.5.7.

The following discussion describes how an initialization file is executed. (Note: since the interactive use of auto-exec is covered in Chapter 9 of the *Interactive System User's Guide*, the discussion here is limited to batch use of auto-exec.)

After an initialization file has been implemented by the PN manager using AUTHORF, individual users under that problem number will be subject to its restrictions, unless a user requests otherwise. The job card parameters INIT and NOINIT allow the user some control over execution of the initialization file.

NOINIT — If specified on the job card, no initialization file will be used, unless the PN manager has made execution of the file mandatory; in this case, the job will abort.

INIT — This causes the initialization file to be used. If the initialization file is not available, or if the PN manager has requested that the file not be used, the job will abort.

The initialization file, cataloged under the permanent file name "INITFILEFORPNnnnnnBATCH" is attached at the start of the job to the local file INITFIL. Processing of the file continues as it meets one of the following conditions:

1. If the file contains a program, the control statement 'INITFIL.' is executed, after which any control statements in the file INPUT are processed.

2. If the file contains a sequence of control statements, those statements will be executed; then control returns to the INPUT file. The initialization file may contain an EXEC statement; the control statements added by that EXEC will be executed before control returns to the INPUT file. Executing an EXIT statement when processing INITFIL will cause control to return to the file INPUT unless 'EXIT,U.' (unconditional termination) or 'EXIT,C.' (continue execution) are specified (see Section 7.1.3).

3. If the initialization file cannot be attached, the job will not be allowed to execute. This prevents a job from executing a control statement sequence that depends on the results of the initialization file.

4. If the initialization begins with an end-of-section (EOS) or an end-of-partition (EOP), the job will abort.

Example 2: Executing an auto-exec file

Suppose that the exec file created in Example 1 above is to be automatically executed when a user submits a batch job which includes data to be merged. The PN manager can create the initialization file by running the following job:

```
PNC
job card
PW=password
COPYCR,INPUT,X.
AUTHORF,CHANGE,BINIT,LFN=X,PW=MRGFILEPW,REQUIRED.
7/8/9
control statements as given in Example 1
6/7/8/9
```

A user need only run the following job to have the control statements in the initialization file process the given data.

```
PNC
job card
PW=password
7/8/9
data
6/7/8/9
```

7.1.3 EXIT

The EXIT statement specifies that a group of control statements is to be executed in case of a fatal job error.

```
EXIT[,option].
```

If 'EXIT.' alone is given in the control section, the following actions will take place: if no error occurs, the job will terminate when 'EXIT.' is encountered. When an execution error occurs (see list below) the control statements following the 'EXIT.' statement are executed.

The options described below cause varied results depending on job error status at the time the 'EXIT,option.' statement is encountered.

S allows processing to continue even after a control statement error. If a control statement error occurs, the control section is searched for an 'EXIT,S.' statement.

C causes execution to continue even if no errors are encountered; if executed normally, control will be passed to the following statement, instead of terminating the job. If an error occurs, 'C' has no effect.

U causes an unconditional exit (applies to batch initialization files only). If 'EXIT,U.' is executed normally, the job will terminate. If an error occurs, 'U' has no effect.

Without the U or C parameter, if an EXIT statement is executed normally from a batch initialization file, control returns to the input file control section. **Note:** C and U may not be specified on the same EXIT statement; either, however, can be used with S.

The following chart summarizes the action caused by the various EXIT statements. "Resume processing" means control is passed to the next control statement; "skip" means control is passed to the next EXIT type statement.

Error Condition	EXIT.	EXIT,C.	EXIT,S.	EXIT,U.	EXIT,C,S.
No error	End job[1]	Resume Processing	End job[1]	End job	Resume Processing
Special errors (see list below)	Skip	Skip	Resume Processing	Skip	Resume Processing
Execution errors (see list below)	Resume Processing	Resume Processing	Resume Processing	Resume Processing	Resume Processing

Special errors:

1. Control statement errors.

2. Attempt to load output from a bad assembly or compilation.

Execution errors:

1. Requested resources exceeded—Job has used all central processor time, money (job cost), files, or mass storage that it requested on the job card.

2. Operator drop—processing of a job step is halted by the operator.

3. Arithmetic error—central processor error exit has occurred; this includes mode errors.

4. PP abort—peripheral processor has encountered an illegal request such as illegal file name or request to write outside the job field length.

5. CP abort—central processor program has requested that the job be terminated.

6. PP call error—monitor has encountered a peripheral processor call error entered in RA+1 by a central processor program.

7. ECS parity error.

8. Auto-recall error—job entered auto-recall with completion bit set.

9. Job hung in auto-recall—no activity exists for a job in auto-recall, and completion bit is not set.

Certain conditions cause immediate termination of a job, regardless of the EXIT statement. These are:

[1] Return to input file of batch job if encountered on an initialization file.

1. return of the job to the input queue (RERUN);

2. a job card error; or

3. incorrect reading of the job (a checksum error during job input).

Note: Any job interrupted because of resource limits (job cost, time, files or mass storage) being exhausted will be given an extension of that resource to allow the user to catalog any intermediate results, list any output, etc. The amount of the extension is equal to the following:

Dollar limit:	one-half of the job card JC value or $1.00, whichever is less.
Time limit:	5 CPU seconds
Disk storage limit:	500_8 PRUs
File limit:	5 files

Example:

The example below shows how 'EXIT.' and 'EXIT,S.' might be used to inhibit unwanted output. Assume that debugging information from program EXAMPLE is to be written on TAPE1.

```
PNC
job card
PW=password
FTN,L=LIST.
LGO.
EXIT.
REWIND,TAPE1.
COPYCF,TAPE1,OUTPUT.
EXIT,S.
ERRS,I=LIST,ALL.
7/8/9
      PROGRAM EXAMPLE(INPUT,OUTPUT,TAPE1)
      .
      .
      .
7/8/9
data
6/7/8/9
```

If the job executes normally, the 'EXIT.' statement will end the control section and only the results of execution will be printed. Should FTN run-time errors, mode errors, time or dollar limits occur, control will skip to the 'EXIT.' statement, and the debugging print on TAPE1 will be copied to OUTPUT; and execution will then terminate. If compilation errors occur, when an attempt is made to load and execute the compiled binary (the LGO statement) control will skip to the 'EXIT,S.' statement and a list of the errors is produced by ERRS (see Section 7.13.4).

7.2 Authorization File Manipulation

The following control statement, which is unique to MSU, examines or alters various fields of the user's Authorization File entry. For a description of these fields, see Section 2.6.2.

7.2.1 AUTHORF

AUTHORF is a utility for manipulating and displaying the contents of the Authorization File. The items that may be altered and/or listed are determined by the user's level of authorization. See Chapter 2 for a complete description of AUTHORF.

The AUTHORF control statement has two forms. The following form would be suitable when only one AUTHORF directive is to be executed:

```
AUTHORF,afdirective.
```

afdirective any legal AUTHORF directive (see Section 2.5).

When more than one AUTHORF directive is to be executed, the following form should be used:

```
AUTHORF[,I=inlfn][,O=outlfn][,ABORT].
```

I=inlfn specifies the name of the file from which AUTHORF is to read the directives. The default is INPUT in batch mode; in interactive mode the directives will be read from the terminal.

O=outlfn specifies the name of the file upon which AUTHORF writes error messages, DISPLAY output, echoed input lines, etc. The default is OUTPUT in batch mode, terminal for interactive mode.

ABORT requests AUTHORF to abort the job in the event of fatal AUTHORF errors. Unless this parameter is specified, AUTHORF will always terminate normally; that is, control will pass to the next control statement regardless of any AUTHORF errors.

Note: Both forms of the AUTHORF control statement allow the processing of continuation cards in batch mode. The parameter list may be separated after any comma and continued on the next card.

Examples: See Chapter 2 for examples, especially Section 2.5.3.

7.3 Job Input

The following statements identify the user to the authorization system.

7.3.1 Job Card

The job card must follow the PNC on all user job decks submitted via card readers, or be the first statement in a job submitted via DISPOSE. The format is shown below. Optional parameters may be specified in any order.

```
id[,Cc][,CMcm][,INIT|,NOINIT][,JCjc][,Ll][,M64][,MSms][,MTmt][,NTnt][,PNpn]
   [,RGrg][,Tt].
```

The parameters are in effect for the current job only.

Each parameter is fully described in Section 3.2.3. Briefly, the parameters' functions are:

1. to specify restrictions on job cost (JC), print limit (L), cards punched (C), central memory usage (CM), execution time (T), and mass storage (MS).
2. to specify the job's tape drive requirements (MT for 7-track tapes, NT for 9-track tapes).
3. to specify a problem number on jobs created by DISPOSE (PN).
4. to specify the rate group (RG).
5. to control initialization file use (INIT, NOINIT).
6. to specify processing on the 6400 (M64).

Examples: See Section 3.2.3 for examples.

7.3.2 Password Card

The password card specifies the password associated with the user ID of the job card, as described in Section 3.2.4. It is the only control statement that may not end with a period or right parenthesis. The Problem Number manager can make the password card a requirement for all jobs submitted via card readers by setting the PW-REQUIRED flag via the AUTHORF utility's CHANGE directive (see Section 2.5.6). Note: the password card should not be used for jobs submitted via DISPOSE.

Section 2.2 includes instructions for changing the password.

Example:

```
PNC
job card
PW=DONOTTELL
.
.
```

7.4 Equipment and File Assignment

The following statements are concerned with requesting, releasing, and describing the status of files. Statements related to various phases of tape processing are also included.

7.4.1 DISPOSE

The DISPOSE statement specifies how a local file is to be processed after it is released from the user's job. According to the disposition specified, the file will either be printed, punched, placed in the input queue and executed as a separate job, or sent to another site of the Merit Network for execution. This statement is not applicable to permanent files or tapes.

The DISPOSE statement has three forms.

Form 1: DISPOSE,lfn,dis[=dest][,C=cpy][,L=lmt][,I=acctlfn].

Form 2: DISPOSE,*lfn,dis.

Form 3: DISPOSE,**,dest.

Forms 1 and 2 are used to dispose specific files to a site, while Form 3 routes an entire job output to a site. Optional parameters for Form 1 specify output site, copies count, and page or card limit.

lfn the local file name (required for Forms 1 and 2). This cannot be a permanent file, but it may be a special file name, such as OUTPUT, PUNCH, or PUNCHB. The file INPUT, however, cannot be processed by DISPOSE.

Form 1 releases the file from the job immediately. Any further reference to this file name will create a new file or cause an error, depending on the purpose of the reference.

Form 2 does not release the file. The file will not be printed or punched until the job terminates or until the file is returned by a separate request. In this form the C=cpy, L=lmt, I=acctlfn, and dest parameters are ignored, and the IN disposition is illegal. The file is given the copies count (see Section 7.12.3) and print/punch limit associated with the job output (specified on the job card). If a destination other than the default is desired, it is necessary to include a 'DISPOSE,**,dest.' statement before the statement that returns the file. For example, a remote batch terminal user who wishes to print a file at the central site after the job has completed, must use the following control statements:

```
DISPOSE,**,B.
DISPOSE,*lfn,PR.
```

dis a disposition code for print or punch files, or for files to be submitted as separate batch jobs. This parameter is required for Forms 1 and 2.

Currently some line printers at the central site can print both ASCII and Display code files, while other printers can print only Display code files. Four different printing dispositions are available:

Code	**meaning**
PAF	Print the file on a 96-character upper/lower case printer only.
PAU	Print the file on a 64-character, upper case ASCII printer only. Any lower case ASCII characters in the file will be mapped to upper case equivalents.
PA	Print the file on any printer that can print an ASCII file (i.e. those described under 'PAF' and 'PAU' above).
PR	Print the file on any available central site printer. Caution: Specifying this disposition code may cause an ASCII file to be printed on a printer that is intended to process only Display code; in such a case the result will be unreadable output.

Punch files may be specified by one of the following codes:

Code	**Meaning**
PU	Punch the file using 026 keypunch codes. Unit records longer than 80 columns are continued on succeeding cards.
PB	Punch the file in standard binary mode.
P8	Punch the file in 80-column binary mode.
P9	Punch the file using 029 keypunch codes. Unit records longer than 80 columns are continued on succeeding cards.
PC	Punch only the first 80 columns of each unit record, using 026 codes.

The following disposition code specifies that the file is to be executed as a separate batch job.

Code	**Meaning**
IN	Place the file in the input queue and execute it as a batch job. If the job is to be executed at the central site, the first unit record of the file must be a legal job card containing the 'PNpn' parameter. If the job is to be disposed via the Merit Network, the first unit record must contain an appropriate job card for the University of Michigan or Wayne State. Each batch job is allowed only one 'DISPOSE,lfn,IN.' statement.

dest an optional destination code for print files. See Appendix E for legal values of dest.

This parameter is optional on Forms 1 and 2, and required on Form 3. If not specified, print returns to the site of job origin or, in the case of interactive system users, to the central site (source B). Only source codes for which the user is authorized may be specified.

Note: Form 3 changes the default destination for subsequent DISPOSE statements in the job.

A job may be disposed for execution via the Merit Network by specifying the disposition (dis) IN and a destination of MS, WU, or UM. Output can be printed or punched via Merit by specifying the PR or PU disposition and the MS, UM, or WU destination code. See Chapter 7 of the *Merit User's Reference Manual* for an example of disposing batch jobs via the Merit Network.

The dest parameter is illegal for jobs disposed with an 'IN' disposition, unless the destination is a Merit host, as described above.

C=cpy the number of **additional** copies desired (0≤cpy≤63). This parameter applies only to print and punch files; it cannot be used with 'IN.' If C=cpy is·not specified, no additional copies will be printed or punched. The total number of pages or cards required for all copies is subject to the print or punch limit currently in effect.

I=acctlfn a local file containing accounting information for jobs disposed to Wayne State or University of Michigan through the Merit Network. This parameter should only be used when disposing jobs through Merit.

L=lmt a card or page limit, used only for print and punch files. If not specified in interactive use, the authorization file maximum page or card limit applies. If not specified in batch, the job card page or card limit is used, or the authorization file default if job card limits are not given.

Forms 1 and 2 cause each print file to be printed with separate banner and trailer pages and a one-line dayfile message giving the print charge; the entire dayfile for the job is printed at the site of job origin. If Form 3 is used all output, including the job dayfile, is printed at the specified site, and the default destination is changed for future DISPOSE statements.

Any output file disposed to print from a batch job is assigned the sequence number of the job; print files disposed from an interactive terminal are identified by the interactive session sequence number. Note that the second character of the sequence number may be different if the output is directed to a different I/O source. For example, 'SSnnnnn' becomes 'SBnnnnn' for print files from interactive terminal sessions. Each file disposed to IN or to the Merit Network from batch or interactive jobs, is assigned a unique sequence number that is displayed when the DISPOSE command is executed. See the *Interactive System User's Guide*, Chapter 7, for a complete description of disposing jobs to the input queue.

When disposing a file to print, the user should make sure that column one of each line contains an appropriate carriage control character. For example, if file lfn$_1$ does not have carriage controls, it can be single spaced by executing 'COPYSBF,lfn$_1$,lfn$_2$.' and then disposing lfn$_2$ to print (see Section 7.5.7).

Example 1: Disposing LGO to punch.

The following jobs illustrate the difference between the lfn and *lfn forms of DISPOSE.

```
a.   PNC                          b.   PNC
     job card                          job card
     PW=password                       PW=password
     FTN.                              FTN.
     LGO.                              DISPOSE,*LGO,PB.
     DISPOSE,LGO,PB.                   LGO.
     7/8/9                             7/8/9
     FORTRAN program                   FORTRAN program
     6/7/8/9                           6/7/8/9
```

The DISPOSE statement in job (a) specifies that the file LGO is to be released immediately and punched in standard binary format. In job (b), LGO is not released until end-of-job. If the asterisk were omitted on the DISPOSE statement in job (b), LGO would be released immediately and the job would abort when the 'LGO.' statement requested to load a nonexistent file.

Example 2: Printing copies of a permanent file.

```
PNC
job card
PW=password
ATTACH,INFILE,DMSPERMFILE.
COPYSBF,INFILE,OUT.
DISPOSE,OUT,PR=A,C=9,L=100.
6/7/8/9
```

Because permanent files cannot be disposed, the permanent file DMSPERMFILE is attached as local file INFILE and copied to a local file named OUT; the COPYSBF statement is used to assure proper carriage controls in the file OUT (see Section 7.5.7). In this example the DISPOSE statement directs file OUT to the printer at source A, the remote terminal in 208 Computer Center. It also requests 10 copies (9 is the number of **additional** copies) and a maximum of 100 pages for the total print.

Example 3: Directing job output to another site.

```
PNC
job card
PW=password
DISPOSE,**,V.
FTN.
LGO.
7/8/9
FORTRAN program
6/7/8/9
```

Assuming that this user is authorized for source V, this job will print at the remote batch terminal in the Engineering Building.

Interactive use of DISPOSE is documented in the *Interactive System User's Guide*, Chapter 7.

7.4.2 FILE

The FILE control statement describes attributes of a file which will be read or written by Cyber Record Manager (CRM). The description remains in effect until the job terminates or until overridden by a subsequent FILE statement.

```
FILE,[lfn][,descriptors].
```

lfn	local file name of the file being described.
descriptors	see the CDC *Cyber Record Manager Reference Manual*, Section 2.

Note: The FILE control statement allows processing of continuation cards in batch mode. The parameter list may be separated after any comma and continued on the next card.

'FILE,lfn.' nullifies all previous FILE statements describing lfn. 'FILE.' erases all previous FILE statements.

The information on a FILE statement is saved on a file named ZZZZZDF.

For languages that do not normally use the full CRM library (such as FTN), the library set must be declared such that the language's library is searched first, then the CRM library. For example:

```
FILE,TAPE1,RT=F,FL=80,BT=C.
FTN.
LDSET,LIB=FORTRAN/CRM.
LGO.
```

When loading absolute (core-image) programs, no LDSET statements are needed. For example,

```
FILE,TAPE10,RT=Z,FL=80,BT=C.
SORTMRG.
```

At execution time, the information on a FILE statement is placed in the FIT (File Information Table) by Cyber Record Manager when the file is opened or when a SETFIT macro is executed.

Caution: Not all programs use Cyber Record Manager for input and output. CRM is used by:

SORT/MERGE (see Section 7.5.12)
FORM (see Section 7.5.10)
DMS-170 utilities (see Section 7.19)
Cyber Record Manager utilities (see Section 7.18)
COBOL version 4 programs (see Section 7.16.3)
FTN version 4 programs (if the CRM library is requested; see the 'LDSET,LIB=' statement, Section 7.9.2).
FTN version 5 programs (see Section 7.16.6).

For a complete description of FILE, see the CDC *Cyber Record Manager Reference Manual* (Publication No. 60495800).

7.4.3 FILES

The FILES utility lists the names of all local files assigned to the user's job. The format of the FILES statement is as follows:

```
FILES[,O=outlfn].
```

O=outlfn   specifies the output file name. From batch, the default is OUTPUT. TTYTTY is the default file for interactive use; if the O parameter is specified alone, output will be on the connected file ZZZZOT.

FILES adds a prefix to the local file name to indicate whether the file is permanent or connected, as follows:

P*lfn	permanent file
C*lfn	connected file (interactive)
lfn	other file

FILES is intended primarily for interactive use, but is also useful when a batch job is aborted for exceeding the authorized file limit. FILES enables the user to display the names of all files so he/she can return some on subsequent runs.

7.4.4 NEWNAME

NEWNAME will change a file's local file name.

```
NEWNAME,oldlfn,newlfn.
```

Both parameters are required.

oldlfn the file's current local file name.

newlfn the file's new local file name.

Legal SCOPE/HUSTLER local file names must consist of a letter followed by 0 to 6 letters and/or digits. If either file name is illegal, or if oldlfn does not exist, NEWNAME aborts with the message "INVALID FILE NAME." If newlfn already exists, NEWNAME aborts with the message "DUPLICATE FILE NAME."

Cautions:

1. Special file names (discussed in Section 4.1.7) cannot be used as either oldlfn or newlfn.

2. NEWNAME should not be used on tape files; the results are unpredictable.

Example:

```
PNC
job card
PW=password
ATTACH,FILE1,PERMFILE1.
ATTACH,OLDPL,PERMFILE2.
UPDATE,Q.
RETURN,OLDPL.
NEWNAME,FILE1,OLDPL.
UPDATE,O=OUTFILE.
6/7/8/9
```

This example illustrates the use of the NEWNAME statement to rename a file in order to cause it to be processed by UPDATE. In UPDATE, the file OLDPL refers to the old program library.

7.4.5 REQUEST

The REQUEST statement requests the assignment of a device for either a disk or magnetic tape file. Because SCOPE/HUSTLER will create a disk file automatically whenever a job references a previously undefined local file name, explicit requests for disk files are not necessary.

MSU supports both 7- and 9-track tapes. Most parameters of the 7-track tape request are common to the 9-track tape request, and the descriptions apply to both control statements. Exceptions are noted where appropriate. **Due to the limited number of tape drives, tape requests from interactive jobs are illegal.** Only batch jobs may use tapes.

```
REQUEST,lfn[,optional parameters].
```

lfn

the local file name to be assigned to the file. If lfn is the only parameter specified, the request is assumed to be for a disk file; otherwise it is assumed to be a tape request.

VRN=vrn[=...]

one or more visual reel names, separated by equal signs. If a volume trailer label (EOV) is encountered, the system will rewind and unload the reel and request the next VRN in the list; if the VRN list is exhausted, the job will abort. Normal termination will occur only if the system encounters a file trailer label (EOF) or an end-of-tape (EOT) reflective strip (see Section 6.11.2). Up to 62 visual reel names can be specified in this list.

If the VRN list is omitted but some other tape parameter is specified, a scratch tape will be assigned. In the interest of efficiency, disk files should be used rather than scratch tapes. There is no way to guarantee the contents of a scratch tape from one run to another, or even to request the same scratch tape.

dns

specifies the density at which the data will be recorded or read, in terms of characters per inch (cpi). This also specifies whether the tape is 7-track or 9-track. The density must always be specified when using a 9-track tape; otherwise the system assumes the tape is 7-track. When a labeled tape is read, the density parameter has no effect because the density is specified in the header label.

The following are legal values for dns:

7-track tapes		9-track tapes	
LO	200 cpi	HD	800 cpi
HI	556 cpi	NT	1600 cpi (phase-encoded)
HY	800 cpi	PE	1600 cpi (phase-encoded)
MT	556 cpi		
omitted	556 cpi		

cconv

character conversion mode (for 9-track stranger (S and L) tapes only); cconv may be one of the following:

AS	perform ASCII character translation
EB	perform EBCDIC character translation
omitted	perform ASCII character translation

rwmode

specifies whether the reel is to be accessed in a read-only or read-write mode.

RW	read and write access to be permitted
RO	read-only; no writing is permitted on the tape
omitted	read-only; no writing is permitted on the tape

fmt

the data format:

S	stranger tape
L	long block stranger tape
omitted	SCOPE tape

lbl specifies the label format. The default is E or N.

E or N	SCOPE-ANSI labels
Z	unlabeled
Y	3600 labels (illegal on 9-track tape request). These labels are obsolete; Y should not be used when writing new tapes.
omitted	SCOPE-ANSI labels

If an unlabeled tape is requested as labeled and the first action is a write, a labeled tape will be created. But if the first action is a read, the system will attempt to check label information and, finding unrecognizable information, will request the operator to either drop the job or allow it to run (see Section 6.11.5). The same action will occur if the wrong label style is specified. If a labeled tape is requested as unlabeled (Z parameter), the tape will be positioned after the label (unless the first action is a write operation).

NS indicates nonstandard label processing. This parameter must be used in conjunction with the Z parameter and the S or L parameter. After the initial label checking is performed, the tape is repositioned before the label. The contents of the label are treated as data.

NB or IB no brackets (inhibit brackets)—suppresses noise bracketing of unusable tape; alternate procedures for recovering write parity errors are used instead. See Section 6.8. Note: NB and IB are ignored on 9-track tape requests since noise bracketing is not used.

NR specifies no read recovery. Data with errors will be transferred to the user's buffer with no error indication.

NEWPN=pn a 6 or 7 character problem number, or zero, indicating the owner of the tape. Subsequent jobs may write on this tape only if they are run under the problem number specified by the NEWPN parameter when the tape was created. This parameter is used only when creating a labeled tape; it has no effect when either the RO or Z parameter is specified. The value specified by pn is written into the PN field of the header label sequence for both SCOPE/ANSI and 3600 (obsolete) labeled tapes. A value of 0 (zero) will permit any user to write on the tape.

If NEWPN=pn is omitted when a labeled tape is created, the problem number from the job PNC is used.

Note: The REQUEST control statement allows processing of continuation cards in batch mode. The parameter list may be separated after any comma or equal sign and continued on the next card.

Examples: See Section 6.5.1 for examples.

7.4.6 RETURN

The control statement RETURN releases files from the user's job and returns them to the system. Permanent files are "detached" from the job; an ATTACH is necessary to access the file again. Temporary files cannot be accessed after they have been returned.

```
RETURN,lfn,[,...][,ntdr].
```

lfn local file name(s) of file(s) to be returned.

ntdr new tape drive reservation. This parameter is meaningful only when returning files stored on magnetic tape. ntdr may be one or both of the following:

```
MT={m|SAME}
NT={n|SAME}
```

MT specifies a value for the job's 7-track tape drive reservation ($0 \leq m \leq 4$); NT specifies a value for the job's 9-track tape reservation ($0 \leq n \leq 4$). The new reservation cannot be greater than that specified on the job card.

If NT is not specified, the job's 9-track tape reservation is decremented by 1 for each 9-track tape file returned; if MT is not specified, the job's 7-track reservation decremented by 1 for each 7-track tape file returned. The 7-track reservation will not be decremented past 1, unless the 9-track reservation is first raised above zero.

Note: RETURN cannot be used to increase a job's tape drive reservations, unless all tape files are returned.

Examples: Examples of RETURN may be found in Section 6.4.3.

7.4.7 TAPRES

The TAPRES control statement specifies the maximum number of tape drives needed by the job at any one time. Its use is subject to the rules listed in Section 6.4.

```
TAPRES[,MT={m|SAME}][,NT={n|SAME}].
```

m the number ($0 \leq m \leq 4$) of 7-track tape drives to be reserved.

n the number ($0 \leq n \leq 4$) of 9-track tape drives to be reserved.

SAME indicates that the corresponding reservation limit is not changed.

Example: See Section 6.4.2 for an example of TAPRES.

7.4.8 UNLOAD

The UNLOAD control statement releases files from a user's job. Unlike RETURN (see Section 7.4.6), UNLOAD has no effect on the job's tape unit reservation. Tape files are rewound and physically unloaded; the tape drive is returned to the system.

Permanent files are "detached" from the job; an ATTACH is necessary to access the file again. Temporary files cannot be accessed after they have been unloaded.

```
UNLOAD,lfn[,...].
```

lfn local file name(s) of file(s) to be unloaded.

Example:

```
PNC
id,NT2.
REQUEST,TAPE1,VRN=100,RW,NT.
REQUEST,TAPE2,VRN=101,NT.
COPYCR,TAPE2,TAPE1.
UNLOAD,TAPE2.
REQUEST,TAPE3,VRN=102,NT.
COPYCR,TAPE3,TAPE1.
6/7/8/9
```

Two 9-track tape drives are reserved, while a total of three tapes are used in the job. After tape 101 (TAPE2) is used, the UNLOAD statement causes it to be unloaded without reducing the tape drive reservation by 1.

7.5 File Copying and Reformatting

The following control statements allow the copying of data from one file to another, or cause the data in a file to be reformatted or rearranged.

7.5.1 COMBINE

COMBINE reads two or more SCOPE sections from one local file and writes them as a single SCOPE section (level 0) on a second local file.

```
COMBINE,inlfn,outlfn,n.
```

inlfn	name of the local file containing sections to be combined.
outlfn	name of the local file to which combined section are to be written.
n	number of sections to be read from inlfn.

All three parameters must be supplied by the user; there are no defaults. The operation terminates when the section count is exhausted or when an end-of-partition is encountered on file inlfn.

Cautions:

1. Files inlfn and outlfn are not repositioned before or after the COMBINE operation.

2. File inlfn cannot be an S or L tape.

3. If files inlfn and outlfn have not been previously defined, they will be assumed to be disk files.

Example:

```
PNC
job card
PW=password
ATTACH,OLDDATA,MYDATAFILE.
SKIPF,OLDDATA.
COPYCR,INPUT,OLDDATA.
COMBINE,OLDDATA,NEWDATA,2.
CATALOG,NEWDATA,MYNEWDATA,RP=30.
PURGE,OLDDATA.
7/8/9
data
6/7/8/9
```

In the example, the permanent file MYDATAFILE, attached as local file OLDDATA, contains one section. Data from INPUT is copied to OLDDATA, then the data is combined into one section on local file NEWDATA. The old file is then purged, and the file containing the combined data is cataloged for future use.

7.5.2 COPY

This control statement causes an input file to be copied onto an output file from the current position up to end-of-information or double end-of-partition on the input file. Both files are then backspaced over the last file mark. No other file positioning is done before or after this operation.

```
COPY,[inlfn],[outlfn].
```

inlfn specifies the name of the input file. The default is INPUT.

outlfn specifies the name of the output file. The default is OUTPUT.

Caution: COPY will give undefined results on stranger (S or L) or coded SCOPE tapes.

Example:

```
PNC
job card
PW=password
ATTACH,ADATA,PERMANENTFILE.
COPY,ADATA,PUNCHC.
6/7/8/9
```

In this example, the user attaches a permanent file to local file ADATA, then copies ADATA to PUNCHC, which is a special output file producing a deck of punched cards.

7.5.3 COPYBCD

COPYBCD copies partitions containing SCOPE records (RT=Z) from an input file in such a manner that each line image is written as a discrete block on the output file. It is useful for copying files to a magnetic tape so the tape may be transported to a site requiring separate blocks for each record.

```
COPYBCD,[inlfn],[outlfn],[n].
```

inlfn specifies the name of the input file. The default is INPUT.

outlfn specifies the name of the output file. The default is OUTPUT.

n specifies the number (decimal) of partitions to be copied from the input file.

Each line of inlfn is assumed to be terminated by a zero byte (RT=Z). The file outlfn is written on a tape in coded mode (BCD; even-parity if the tape is 7-track) with each line a block consisting of 148 characters with the zero byte converted to blanks.

When an end-of-partition is encountered on the input file, a printer carriage control character is written on the output file before an end-of-partition is written; this forces a skip to the top of the next page. Thus, if listed on a printer, the output begins each file at the top of a new page. Extraneous characters appear at the top of a new page as a result of the skip and end-of-partition marks on the tape.

Example:

This example shows how a permanent file is copied to a magnetic tape in the format described above.

```
PNC
id,MT1.
PW=password
ATTACH,X,PERMANENTDATA.
REQUEST,TAPE1,VRN=101,S.
COPYBCD,X,TAPE1.
6/7/8/9
```

7.5.4 COPYBR, COPYCR, COPYBF, COPYCF

Individual partitions or sections in a file may be copied by the control statements described in this section. A distinction is made here between "coded" data and "binary" data. This distinction is meaningful only when applied to data being copied to or from magnetic tapes. Copying binary sections or partitions causes the data to be in exactly the same form as it is represented in central memory. When coded mode is used, a translation process takes place; this process is described in Section 6.8.8.

COPYBR copies a specified number of binary sections from one file to another file.

```
COPYBR,[inlfn],[outlfn],[n].
```

COPYCR copies a specified number of coded sections from one file to another file.

```
COPYCR,[inlfn],[outlfn],[n].
```

COPYBF copies a specified number of binary partitions from one file to another file.

```
COPYBF,[inlfn],[outlfn],[n].
```

COPYCF copies a specified number of coded partitions from one file to another file.

```
COPYCF,[inlfn],[outlfn],[n].
```

inlfn specifies the name of the file that contains the data to be copied. The default is INPUT.

outlfn specifies the name of the file to which the data is copied. The default is OUTPUT.

n specifies the number (decimal) of sections or partitions to be copied. The default is 1.

COPYBR and COPYCR have identical effects when copying disk files. When copying magnetic tape files, however, care should be taken that COPYCR is used to copy data on coded tapes and COPYBR is used for binary tapes. Likewise, COPYCF and COPYBF have identical effects on disk files; COPYCF should be used for coded tapes, and COPYBF for binary tapes. See Section 6.8.8 and 6.8.9 for a discussion of binary and coded tapes.

These copy routines open the files specified.

COPYBF and COPYCF terminate when the requested number of partitions are copied or when an end-of-information is encountered on inlfn.

COPYBR and COPYCR terminate when the requested number of sections are copied or when an end-of-partition is encountered on inlfn. The operation will cease (but not abort) if an end-of-partition is encountered on inlfn before the section count is exhausted; an appropriate message is entered in the dayfile. An end-of-partition is then written on outlfn and backspaced over, and the file remains open. If an end-of-information is encountered on inlfn before the section or partition count is exhausted, the operation will cease (but not abort). An appropriate message is then entered in the dayfile, an end-of-partition is written on the file outlfn, and both files are closed.

Cautions:

1. These copy routines cannot be used to copy random files. They should not be used to copy EDITOR work files; the user is advised to use 'HAL,FCOPY.' (see Section 7.5.9) for such an operation.

2. These copy routines cannot be used on connected files (interactive mode). Instead, use the COPY control statement (see Section 7.5.2), or READPT or WRITEPT (see Chapter 2 of the *Interactive System User's Guide*).

3. When COPYCF or COPYCR is used to copy files with more than one line per section to a stranger (S) tape, the line images (records) will not be interpreted as separate sections.

4. These copy operations normally require 11300 (octal) words of memory to execute. However, when copying long-block (L) stranger tapes, the user should specify in addition at least twice the length of the longest block on the tape to accommodate increased buffer sizes. 'AUTORFL,OFF.' should be specified immediately before the copy control statement to allow the routine to take advantage of its additional memory. (**Note:** 'AUTORFL,OFF.' should be used with caution; see Section 7.11.1.)

Examples:

The following example illustrates the use of COPYCR to copy the first five sections of a coded tape to another tape, and COPYBR to perform a similar operation on a binary tape.

```
Using coded tape:            Using binary tape:

PNC                          PNC
id,MT2.                      id,MT2.
PW=password                  PW=password
REQUEST,A,VRN=123.           REQUEST,A,VRN=123.
REQUEST,B,VRN=234.           REQUEST,B,VRN=234.
COPYCR,A,B,5.                COPYBR,A,B,5.
6/7/8/9                      6/7/8/9
```

To obtain a printed copy of the contents of a tape, the following job could be run:

```
Using coded tape:            Using binary tape:

PNC                          PNC
id,MT1.                      id,MT1.
PW=password                  PW=password
REQUEST,A,VRN=123.           REQUEST,A,VRN=123.
COPYCF,A,OUTPUT.             COPYBF,A,OUTPUT.
6/7/8/9                      6/7/8/9
```

7.5.5 COPYL and COPYLM

The COPYL and COPYLM statements are used to replace selected routines on a binary file. This operation involves three files: oldlfn, the file containing the old set of routines; modlfn, the file containing the replacement routines; and newlfn, the file on which COPYL or COPYLM will place the updated set of routines.

COPYL and COPYLM differ only in the handling of multiple occurrences of a section on oldlfn. COPYL uses each section on modlfn only once, replacing the first matching section from oldlfn. COPYLM uses the first matching section encountered on modlfn to replace each matching section from oldlfn. COPYL can be used for multiple replacement only if multiple copies of the section are on modlfn.

```
COPYL,[oldlfn],[modlfn],[newlfn],[lastsec],[opt].

COPYLM,[oldlfn],[modlfn],[newlfn],[lastsec],[opt].
```

The parameters are optional but order-dependent.

oldlfn the name of the file to be updated. The default is OLDLIB.

modlfn the name of the file containing replacements for selected routines in oldlfn. The default is BINARY.

newlfn the name of the file on which the updated set of routines is to be written. The default is NEWLIB.

lastsec the name of the last section on oldlfn to be processed. If last is not specified, all sections on oldlfn are processed.

opt processing option; the default is to not select an option.

A append to the end of newlfn all modlfn sections that do not match any on oldlfn.

E read the oldlfn until end-of-information rather than end-of-partition.

R rewind oldlfn and newlfn before processing. (Modlfn is always rewound before and after processing.)

T omit check for matching type of section.

These options can be specified by combining one or more letters in any order, such as TAER, AR, RT, or TR.

The routines on file modlfn do not have to be in any particular order; COPYL and COPYLM will scan this file and make an index of the program names. Then, as they copy from file oldlfn tc newlfn, they will insert the routines from modlfn in place of any duplicates found on the file oldlfn. At the end of the operation, file newlfn will contain the subprograms in the same order as they appeared on file oldlfn.

COPYL and COPYLM normally process oldlfn until end-of-partition. The E option allows the user to process oldlfn until end-of-information.

Cautions:

1. Files oldlfn and newlfn are not rewound before or after the COPYL or COPYLM operation unless R is specified in the flag parameter.

2. COPYL and COPYLM will not add routines; they will act only on routines found on both modlfn and oldlfn unless A is specified in the flag parameter.

Example 1: Replacing subprogram TABULAT using COPYL.

```
PNC
job card
ATTACH,OLDPL,JUNKPL.
UPDATE.
COMPASS,I=COMPILE.
REWIND,LGO.
ATTACH,OLD,MYLGO.
COPYL,OLD,LGO,NEW.
CATALOG,NEW,MYNEWLGO.
7/8/9
*IDENT TAB1
 .
 .UPDATE directives
 .
*COMPILE TABULAT
6/7/8/9
```

This job uses UPDATE to modify subprogram TABULAT, whose source language deck is maintained on JUNKPL. UPDATE makes the indicated corrections and copies the deck to COMPILE, which is then assembled into relocatable object form on LGO. After execution of COPYL, the set of decks written to local file NEW will contain the version of TABULAT obtained from LGO.

Example 2: the E parameter in the COPYL statement.

Suppose that OLDLGO contains three programs as follows:

```
Main Sub Sub E Main Sub Sub E Main Sub Sub E E
 X    A   C  O  Y    B   D  O  Z    E   A  O O
             P              P              P I
```

And suppose that replacement decks for subprogram A and B are on LGO:

```
Sub Sub E
 A'  B' O
        P
```

The effect of the COPYL statement depends upon whether or not the E parameter is specified.

a. COPYL,OLDLGO,LGO,NEWLGO,E.

b. COPYL,OLDLGO,LGO,NEWLGO.

If control statement (a) is used, the first copy of A and B on OLDLGO will be replaced on NEWLGO, as shown below.

```
Main Sub Sub E Main Sub Sub E Main Sub Sub E  E
 X    A'  C  O  Y    B'  D  O  Z    E   A  OO
             P           P              P I
```

If control statement (b) is used, NEWLGO will contain:

```
Main Sub Sub E  E
  X   A'  C  OO
             P I
```

Example 3: The COPYLM statement

If COPYLM,OLDLGO,LGO,NEWLGO,E. is specified, the first copy of A and B on OLDLGO will be replaced on NEWLGO as shown.

```
Main Sub Sub E Main Sub Sub E Main Sub Sub E  E
 X    A   C  O  Y    B   D  O  Z    E   A  O O
             P           P              P  I
```

And suppose that replacement decks for subprogram A and B are on LGO:

```
Sub Sub E
 A'  B' O
        P
```

7.5.6 COPYN

COPYN is a utility for consolidating and merging files. Specified sections or binary routines from up to ten input files may be extracted and written into a single output file. COPYN cannot accept S or L tapes as either input or output files.

Directives associated with the COPYN routine determine the final order of the output file. Sections to be extracted may be specified by either name or number relative to current file position. If a section has a prefix table associated with it, the name refers to the name in the prefix table; otherwise the name is taken as the first seven characters of the first word of the section.

The control statement is of the form:

```
COPYN,fmt,outlfn,inlfn[,...].
```

fmt — indicates the section format of the output file; fmt is in the form of a decimal number. If fmt is non-zero and the section to be extracted has a prefix table associated with it, the prefix table will be omitted from the output file. A zero (0) value of fmt indicates the sections are to be copied verbatim.

outlfn — is the local file name of the file to which the specified sections are to be copied.

inlfn — specifies the name(s) of the input file(s). A maximum of ten (10) files may be given.

COPYN reads one section of directives from INPUT. Directives that may be used with COPYN are REWIND, SKIPF, SKIPR, WEOF, and section identification directives. The directives are free field; they may contain blanks, but must include the separators indicated in each card description.

COPYN Directives

REWIND(lfn)

This directive causes file lfn to be rewound. File lfn must be one of the input or output files named on the COPYN control statement. File lfn may not be the file INPUT.

SKIPF(lfn,±n)

This directive causes n partitions on tape file lfn to be skipped forward (+n) or backward (-n). No indication is given when SKIPF causes a tape to attempt to go beyond EOI or before BOI.

SKIPR(lfn,±n)

With this directive, n sections may be skipped forward (+n) or backward (-n) on tape file lfn. Zero length sections and partitions must be included in n. Requests for other types of files are ignored.

WEOF(lfn)

This directive causes an end-of-partition to be written on file lfn, which must be an input or output file named on the COPYN control statement.

Section Identification Directive

The section identification directive contains the following parameters which identify a section or set of sections to be copied from a given file.

p1,p2,p3

p1 is the section to be copied or the beginning section of a set. The name associated with the section or a number giving the position of the section relative to the current position of the file may be specified.

p2 is the last section to be copied in a set of sections. It may be of the form:

name	Sections p1 through p2 are copied.
decimal integer	Number of sections to be copied, beginning with p1. Zero length sections and file marks are counted.
*	p1 through an end-of-partition mark are copied.
**	p1 through a double end-of-partition mark are copied.
/	p1 through a zero length section are copied.
0 or blank	Only p1 is copied.

p3 Input file to be searched. If p1 is a name and p3 is omitted, all input files declared on the COPYN statement are searched by the end-around method described below until the p1 section is found. If p1 is a number and p3 is omitted, the last input file referenced on the COPYN statement is assumed. If this is the first directive, the first input file on the COPYN statement is used.

Section identification directive examples:

SIN,TAN,INPUTA	Copies all sections from SIN through TAN from file INPUTA.
SIN,10,INPUTA	Copies the section SIN and the following 9 sections from file INPUTA.
SIN,TAN	Searches all input files beginning with the current file for SIN. When SIN is found all sections from SIN through TAN are copied (unless an EOP is encountered).
1,TAN,INPUTA	Copies the current section through TAN from INPUTA.
1,10,INPUTA	Copies 10 sections, beginning with the current section from file INPUTA.

Files manipulated during a COPYN operation are left in the position indicated by the previously executed directive. The file containing p1 will be positioned at the section following p2. Other files will remain in the same position.

When COPYN is searching for a named section and p3 has been omitted, each input file is searched in turn until either the named section is found or the original position of the file is reached; this is called an *end-around search*. For example, the statement:

```
SIN,TAN
```

will cause each input file to be searched for SIN, each search beginning at the current point in the file. If end-of-partition is encountered before SIN is found, the file will be rewound and the search continued until the file is positioned at the original point again. An exception is the file INPUT which is not searched end-around, that is, searching terminates at end-of-partition.

In contrast to the end-around search, the copy operation will be terminated when an end-of-partition is encountered even if named section p2 has not been encountered.

COPYN does not check for sections duplicating names on other files. COPYN will use the first section encountered that matches the name on a directive.

An incorrect request results in the abort flag being set and a message being printed on the file OUTPUT followed by the directive in error. This directive is not processed, but an attempt is made to process the next directive. When the last directive is processed, the abort flag is checked; if it is set, the job is terminated. Otherwise, the next control statement is executed.

7.5.7 COPYSxx

COPY Sxx is a general statement format that refers to a group of control statements used to format a file that is to be listed on a printer. The control statement selected depends on whether the file is in ASCII Fancy or Display code format, whether the file is in binary or coded mode (magnetic tape files), and whether you want to copy sections (level 0) or partitions (level 17_8). The table below indicates each flagword and file type.

Flagword	ASCII-Fancy/ Display Code	Binary/ Coded	Section Level	
COPYSAF*	AF	B	17	(partition)
COPYSAP	AF	B	17	(partition)
COPYSAR*	AF	B	0	(section)
COPYSAS	AF	B	0	(section)
COPYSBF*	DC	B	17	(partition)
COPYSBP	DC	B	17	(partition)
COPYSBR*	DC	B	0	(section)
COPYSBS	DC	B	0	(section)
COPYSCF*	DC	C	17	(partition)
COPYSCP	DC	C	17	(partition)
COPYSCR*	DC	C	0	(section)
COPYSCS	DC	C	0	(section)

Note: The flagwords indicated by asterisks adhere to the obsolete terminology of record (meaning section) and file (meaning partition). They are included as legal flagwords to maintain compatibility with other control statements, such as COPYCF and COPYCR.

The calling sequence of each of the COPYSxx statements is:

```
COPYSxx,[inlfn],[outlfn][,n].
```

inlfn specifies the name of the input file. The default is INPUT.

outlfn specifies the name of the output file. The default is OUTPUT.

n specifies the number of partitions or sections to be copied. The default is 1.

A number of sections or partitions (specified by n) are copied from file inlfn to file outlfn with each line shifted right one character. A carriage control character forcing a page eject (the 'I' carriage control) is inserted as the leading character of the first record of each partition or section; a blank is inserted on all succeeding lines, so that the output is single spaced.

Note: You must use the DISPOSE control statement (see Section 7.4.1) when listing ASCII Fancy files on a line printer.

Batch example:

```
PNC
job card
PW=password
ATTACH,X,PERMANENTDATA.
COPYSBF,X,OUTPUT.
6/7/8/9
```

This job produces a printed listing of the contents of the first partition of the permanent file PERMANENTDATA in a single spaced format. The data on file PERMANENTDATA are assumed to be in Display code format. On disk, the binary or "coded" designation is irrelevant.

Interactive example:

```
.
.
.
TPREAD,PTAPE,AF.
 read paper tape
REWIND,PTAPE.
COPYSAP,PTAPE,PRINTUL.
DISPOSE,PRINTUL,PAF.
.
.
.
```

This job reads a paper tape in ASCII Fancy format, and produces a listing of PRINTUL on a central site printer with ASCII Fancy capability. See the *Interactive System User's Guide*, Chapter 7, for more information on disposing output to a line printer from an interactive terminal.

7.5.8 COPY8P

COPY8P is a utility program used to copy IBM 360/370 print files to Control Data-compatible print files. Upper and lower case character capability is maintained, as loss of 8-bit significance is avoided. (Note: upper/lower case print can be obtained by disposing the file to PAF; see Section 7.4.1.) IBM options with respect to record type, block type and print format selection are available. COPY8P does not use Record Manager in copying the print files; therefore, no FILE statements are needed in conjunction with this program.

The control statement is as follows:

```
COPY8P,inlfn,outlfn[,optional parameters].
```

inlfn — local file name of the input file containing 8-bit ASCII or EBCDIC data in IBM 360/370 format. Under normal conditions, the file is on magnetic tape made available to the job by a REQUEST statement.

outlfn — local file name of the output tape or disk file to contain data in a format suitable to the printer.

BLKSIZE=n — defines the maximum block length in 8-bit characters. The parameter n is a decimal count; the default is 137. This parameter has the same meaning as the equivalent 360/370 JCL specification.

CODE=A
CODE=C — defines the character set code present on the input file. A means ASCII; C means EBCDIC. C is the default.

FMT=opt — defines the print spacing convention to be used; opt may be one of the following:

1 single space

2 double space

3 triple space

A first character of each line image is assumed to contain a format control character, as described in the CDC *8-bit Subroutines Reference Manual,* Chapter 5. A is the default.

M first character of each line image is one of the IBM 1403 printer hardware control characters as described in the CDC *8-bit Subroutines Reference Manual,* Chapter 5.

FOLD causes output to be folded to a 64-character set, 6-bit character representation, for printing. Special characters not having a 64-character set representation will not be printed. If omitted, upper and lower case information is preserved for printing; this is the default. Output must be disposed to an ASCII printer if FOLD is omitted.

LRECL=n. defines the maximum record size in 8-bit characters. The parameter n is a decimal count. This parameter has the same meaning as the equivalent 360/370 JCL specification. If omitted, LRECL is assumed to be the same as BLKSIZE; this is the default.

RECFM=rf decribes the record format of the input file. Values for rf are F, V, U, FB, and VB. They have the same meaning as the equivalent 360/370 Job Control Language (JCL) specification. The default is U.

For more complete documentation, and examples, see the CDC *8-Bit Subroutines Reference Manual* (Publication No. 60359400).

7.5.9 FCOPY

FCOPY is a copy routine designed for exact copy features. It copies a file exactly (appending no extra section or partition marks); thus end-of-partition marks (EOP) are introduced only if the input file contains them. Files that end with EOI, not EOS or EOP, can be copied correctly. Data from connected files are combined into one section unless *EOR separates them. FCOPY can also be used to copy random input files, such as EDITOR work files, and to create random output files.

There are two forms of the FCOPY control statement; they are:

Form 1: HAL,FCOPY,[inlfn[/opt]],[outlfn[/opt]],[nsec],[nlev].

Form 2: HAL,FCOPY[,I=inlfn[/opt]][,O=outlfn[/opt]][,R=[nsec][/nlev]].

The syntax of Form 1 is consistent with that of other copy routines on the SCOPE/HUSTLER operating system; the parameters are order-dependent. With Form 2 the parameters are order-independent.

inlfn the name of the input file. The default is INPUT. The file inlfn can be followed by '/opt', described below.

outlfn the name of the output file; the default is OUTPUT. The file outlfn can be followed by '/opt', described below.

/opt can be one or more of the following:

R	rewind the file before copying
NR	do not rewind the file before copying
RND	the file is random
NRND	the file is not random
BIN	the file is a binary file
BCD	the file is a coded file.
WA	the file is word-addressable (applies to outlfn only). The WA option on outlfn drops off the last EOS or EOP (leaving EOI).

The defaults are as follows: The files inlfn and outlfn are binary (BIN); inlfn is rewound (R) before copying and outlfn is not rewound (NR) before copying. Neither file is rewound after copying is complete. The file outlfn is random (RND) by default if the file inlfn is random (RND), even if RND is omitted.

nsec number of sections (decimal) of level nlev to be copied; the default is to copy to end-of-information.

nlev section level number; 0≤nlev≤17 (octal). The default is level 0.

For an example of the use of FCOPY, see Section 6.15.4.

7.5.10 FORM

FORM is a file management utility callable only by control statements. It is not callable by a user program. FORM operates on files or sections supplied by the user. FORM functions can be used on these sections to produce up to 20 output files. FORM's capabilities include: input file description, output file description, nonstandard tape label processing, qualifying records for output, section reformatting, section sequencing, record formatting for printing, and converting System/360 binary or formatted tape files to or from CDC files.

The control statement for calling FORM follows:

```
FORM[,OWN=[lfn|epn]][,I=inlfn][,L=listlfn].
```

OWN=lfn
OWN=epn specifies the file or entry point name of a routine on a library that FORM will load to obtain the binary decks for owncode routines. If a file name is given, one of the entry points on the file must be the same as the file name. If OWN=epn is given, the LDSET loader control statement must be used to define the library containing epn (see Section 7.9.2). If this parameter is omitted, it is assumed any owncode routines referenced by the XEQ directive have been loaded by the user. The default is OWN=LGO.

I=inlfn specifies the local file name of the input file containing the directives in card image format (Record Manager record type W with I blocking). The default file is INPUT. FORM does not rewind this file before or after use. Special file names INPUT, OUTPUT and PUNCH have RT=Z and BT=C as defaults for record and blocking type.

L=listlfn specifies the name of the file to receive summary of the FORM run, including diagnostic messages. The default is OUTPUT.

Further documentation, and examples, are found in the CDC *FORM Reference Manual* (Publication No. 60496200).

7.5.11 PAPERT

PAPERT reads paper tapes onto a standard SCOPE file. The control statement has the form:

PAPERT[,optional parameters].

B If specified, will cause the data to be written onto outlfn as SCOPE sections with a maximum section length of 512 words. This produces a file that can be read by the FORTRAN BUFFER IN statement. The default is to copy each tape to file outlfn as a single section.

ID=tapeid A 1-7 character visual identification for the paper tape or set of tapes. This identification is not punched on the tape but should be written on or fastened to each tape (with continuation reel number as well if more than one tape is submitted). The default for tapeid is SCRATCH.

O=outlfn The local file name to which the paper tape is copied. It is not rewound before or after the copy. The default for outlfn is FILE.

T=eot An octal number corresponding to the end-of-tape mark which is placed on the output file at the end of each reel. If eot is a number from 0 to 377_8, the copy operation will terminate when a frame of the tape containing the corresponding character is read. If eot is between 400_8 and 3777_8 or no frame of the tape contains the eot character, the tape will run to the end of the reel and a warning message will be placed in the dayfile. If eot is 4000_8 the leader character, defined as the first character read from the tape, will be used as the eot character. The default is to read to the physical end-of-tape (T=4000).

This utility reads standard 7-row paper tape rolls (fanfold paper tape does not read properly). Each frame is placed in a 12-bit byte, right-justified, zero-filled, and packed 5 bytes per 60-bit word. At the end of each tape, the eot mark is placed in the next byte, and any remaining bytes are filled with 4000_8.

The rolls of paper tape and the job deck to process them should be submitted through the Service Window in Room 208 Computer Center.

7.5.12 SORTMRG

The SORTMRG control statement initiates execution of the system SORT/MERGE utility program. Files are sorted and/or merged based on specifications provided by a section of SORT/MERGE directives.

SORTMRG[,optional parameters].

I=inlfn specifies the file containing the SORT/MERGE control directives. If this parameter is omitted, file INPUT is assumed. If only I is given, the file COMPILE is assumed.

MO=n — specifies the intermediate merge order; $2 \leq n \leq 64$. If insufficient memory is available to merge at the requested order, a fatal error occurs, and a diagnostic indicates how much additional memory would be required. If n is omitted, SORT/MERGE computes a merge order based on the amount of memory available.

O=outlfn — specifies the file on which the output listings will be written. These listings include the directives, dayfile messages, and diagnostics. If O is given alone, or if it is omitted, the file OUTPUT is assumed.

OWN=binlfn — specifies the file on which the OWNCODE binaries are located. If OWN alone is given, the file is assumed to be LGO. If the parameter is omitted, the file is INPUT.

/R or /NR — For I, O, and OWN parameters, if the /R suffix is used (I=inlfn/R,O=outlfn/R, or OWN=binlfn/R), SORT/MERGE will rewind the file when it is opened. If the /NR suffix is used (I=inlfn/NR, O=outlfn/NR, or OWN=binlfn/NR), the file will not be rewound. The default is /NR, no rewind.

6C or 7C — specifies format of SORT/MERGE directives. 6C indicates SORT/MERGE directives are in the formats that were used by SORT/MERGE Version 3.

7C indicates SORT/MERGE directives are in SORT/MERGE Version 4 format. This is the default.

Example: Examples, and further documentation, are found in the CDC *SORT/MERGE Reference Manual* (Publication No. 60497500).

7.6 File Examination

The control statements in this section allow the printing of information regarding a user's local files, permanent files, and magnetic tapes.

7.6.1 COMPARE

The purpose of the COMPARE control statement is to determine whether the contents of two files are identical. COMPARE may be used to verify a tape copy operation.

```
COMPARE,lfn₁,lfn₂,[nsec],[lev],[wdpsec],[maxerr].
```

All parameters are position-dependent. If an optional parameter is omitted, it must be replaced by a null parameter (see examples below).

lfn$_1$,lfn$_2$ local file names of the two files to be compared. Files are not rewound before or after the comparison.

nsec decimal number of SCOPE sections in lfn$_1$, to be compared to lfn$_2$. The default is 1.

lev SCOPE end-of-section level number in octal, without a "B" suffix. End-of-sections greater than or equal to lev are included in the count of nsec sections. The default is 0.

wdpsec decimal number of non-matching word pairs per section to be listed on the file OUTPUT. The default is 0.

maxerr maximum number (decimal) of non-matching sections to be processed. If more than maxerr non-matching sections are found, the comparison is stopped immediately; files are left in their current position. The default is 30000.

Note: If one file is on a coded tape and the other is on disk, the coded tape must be the first file mentioned (lfn$_1$).

The dayfile will include the message GOOD COMPARE if the compared portions of the file are identical. Otherwise, the message BAD COMPARE will appear.

COMPARE will cause information about the failure to match to be written on the file OUTPUT. The quantity of the information is limited by the wdpsec and maxerr parameters. This output may include the following:

Section number (counting from 1) of non-matching sections.

Word number (counting from 0) within a section of non-matching words. This is followed by the octal representation of the word as it appears on each file.

Discrepancy in length between corresponding sections of the two files.

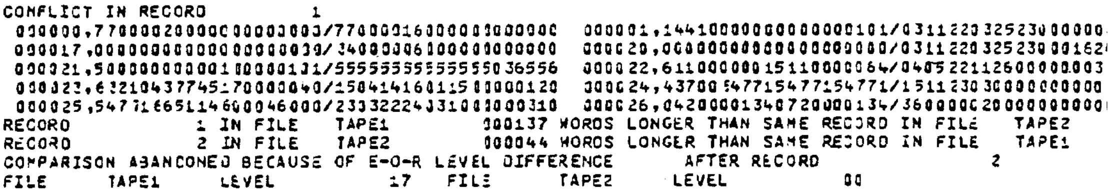
Discrepancy in end-of-section level number between corresponding sections.

Note: Because of the volume of information printed, interactive users should disconnect OUTPUT before doing a COMPARE.

Example 1:

In this example two tape files are compared, and not more than 10 non-matching pairs are to be listed on OUTPUT. The output for this example is shown in Figure 7.1.

```
PNC
id,MT2.
PW=password
REQUEST,TAPE1,VRN=100.
REQUEST,TAPE2,VRN=101.
COMPARE,TAPE1,TAPE2,2,,10.
6/7/8/9
```



```
CONFLICT IN RECORD          1
 000000,770000200000C00003003/7700001600000300000C  000001,14410000000000000101/031122032523000000
 000017,00000000000000000039/3400000610000000000000  000020,00000000000000000000/031122032523000162
 000021,500000000000100000131/55555555555555036556  000022,61100000015110000064/04052211260000000003
 000023,63210437745170000040/15041416011500000120  000024,43700547715477154771/151123030000000000
 000025,54771665114600046000/23332224031000000310  000026,04200001340720000134/36000002000000000000
RECORD              1 IN FILE   TAPE1          000137 WORDS LONGER THAN SAME RECORD IN FILE   TAPE2
RECORD              2 IN FILE   TAPE2          000044 WORDS LONGER THAN SAME RECORD IN FILE   TAPE1
COMPARISON ABANDONED BECAUSE OF E-O-R LEVEL DIFFERENCE       AFTER RECORD                    2
FILE      TAPE1      LEVEL             17    FILE       TAPE2      LEVEL             00
```

Figure 7.1: COMPARE Output (Example 1)

Example 2:

This job compares the first five partitions (level 17 sections) of files A and B.

```
PNC
job card
PW=password
ATTACH,A,PF1.
ATTACH,B,PF2.
COMPARE,A,B,5,17.
6/7/8/9
```

7.6.2 FILEDMP

FILEDMP is a utility that dumps selected full or partial sections from disk or tape files and prints them in any of seven data formats. These formats are described below under "Print Options." The calling sequence for FILEDMP is:

```
HAL,FILEDMP,[miscparams],[setupparams],[actionparams].
```

FILEDMP processes its parameters from left to right in sequence; therefore the order of the parameters is very important. FILEDMP parameters fall into three groups: miscellaneous, setup, and action. The parameters are described in the relative sequence that they should be specified.

Miscellaneous Parameters

Miscellaneous parameters should appear before all other parameters.

I=inlfn — specifies the local file name of the disk or tape file to be dumped. The default is DMPTAP.

O=outlfn — specifies the local file name of the file on which FILEDMP is to write its output. The default is OUTPUT.

Setup Parameters

Setup parameters appear before action parameters, and instruct FILEDMP how to dump succeeding information.

BCD=datafmt — informs FILEDMP that the following information is in BCD (coded tape, even parity) and is to be dumped according to the specified data format (see "Print Options" below). If datafmt is not specified, PR is assumed.

BIN=datafmt — informs FILEDMP that following information is in binary (odd-parity tape) and is to be dumped with the data format option specified by datafmt (see "Print Options" below). If datafmt is not specified, O is assumed.

CC=x — carriage control change. Normally each section dumped is preceded by three blank lines; however, if x is given, it is used as the carriage control character for the first line of each section. Any character other than blank may be specified; see Appendix F for the carriage control action caused by each character.

MNR=n — the first n (decimal) sections of level RL (see below) or less in each partition will be printed. When this limit is reached, FILEDMP skips to the next partition. The default is 131071.

RL=nn — only sections of level nn (octal) or less will be counted ($0 \leq nn \leq 17$). This affects the processing of the MNR, R, and SR parameters. The default is 0.

SUP — instructs FILEDMP to suppress the section level and length messages normally printed between sections. A second occurrence of SUP within the parameter list will restore the printing of these titles. Use of this feature is suggested when dumping stranger (S) tapes.

WC=n — only the first n (decimal) words (10 characters per word) in each succeeding section are dumped. Dumping of sections longer than n is stopped when n is reached. Sections shorter than n are not affected. WC=0 returns the default limit (2^{48}-1).

Print Options

Following is a list of keywords that are legal values for 'datafmt', along with descriptions of the output they produce. These keywords can also stand alone as setup parameters.

FP — instructs FILEDMP to dump data using a floating point format.

H — instructs FILEDMP to dump data in Hollerith format.

HEX — instructs FILEDMP to dump data in hexadecimal format.

I instructs FILEDMP to dump data using an integer format.

M instructs FILEDMP to dump data in COMPASS mnemonics.

O general-purpose format; instructs FILEDMP to dump data 4 words per line in both octal and Hollerith.

PR instructs FILEDMP to search for zero-byte line terminators and dump each line found in Hollerith format.

Action Parameters

Action parameters direct FILEDMP to take immediate action, e.g. to immediately position the file or to immediately dump information in a previously specified format.

CONTINUE directs FILEDMP to continue processing parameters on the next control statement.

F=n directs FILEDMP to dump the next n partitions. If n is not given, 1 is assumed.

R=n directs FILEDMP to dump the next n sections of level RL or less or until a section with level greater than RL is reached. If n is not given, 1 is assumed.

REW directs FILEDMP to rewind the input file. Note: the output file is never rewound.

SF=n directs FILEDMP to skip forward n partitions (level 17_8 sections). If n is not given, 1 is assumed.

SR=n directs FILEDMP to skip forward n (decimal) sections of level RL. If n is not given, 1 is assumed.

Note: The 'HAL,FILEDMP.' control statement allows processing of continuation cards in batch mode. The parameter list may be separated after any comma and continued on the next card.

Example 1:

```
PNC
id,MT1.
PW=password
REQUEST,STAPE,VRN=LY009.
HAL,FILEDMP,I=STAPE,SUP,BCD=PR,F.
6/7/8/9
```

In this example, all section length and level messages are suppressed. The file STAPE is read in coded mode and the first partition is dumped in Hollerith format.

Example 2:

```
PNC
id,MT1.
PW=password
REQUEST,Y,VRN=1002.
HAL,FILEDMP,I=TAPE1,BCD=H,R=1,SR=1,BIN=O,R=1.
6/7/8/9
```

```
SECUENTIAL FILE TAPE1   IS ON SCOPE TAPE        SCOPE LABEL  HI  DENSITY DISPOSITION PUNCH          03/19/79  .14.34.21.


000001        PROG RAM EXAMPL E(INPUT,OU TPUT)                                                     100
000011 C  THIS PR OGRAM IS S UPPOSED TO  READ IN 1 0 NUMBERS  AND                                  110
000021 C  PRINT O UT THE TOT AL AND AVE RAGE OF TH E NUMBERS.                                      120
000031        DIME NSION A(10 )                                                                    130
000041        I=1                                                                                  140
000051        TOTA L=0.                                                                            150
000061 10     READ  100, A(I)                                                                      160
000071 100    FORM AT(F10.2)                                                                       170
000101        TOTA L=TOTAL+A( I)                                                                   180
000111        I=I+ 1                                                                               190
000121        IF (  I.GT.10)  GO TO 20                                                             200
000131        GO T O 10                                                                            210
000141 20     AVER AGE=TOTAL/ 10.                                                                  220
000151         PRIN T 200,TOTA L,AVERAGE                                                           230
000161 200    FORM AT(* *,2F1 0.2)                                                                 240
000171        END                                                                                  250

EOR OF LEVEL 000 FOUND,  CODED  RECORC HAS   1240  0002008 WORDS.
DUMPING TERMINATED AT LEVEL 008.        1 RECORDS DUMPED.

SKIPPING TERMINATED AT LEVEL 008.        1 RECORDS SKIPPED.

000001  77000016000000000000 05300115101405000000 33365034415042445555 34375734415737365557  .;   N        EXAMPLE    03/16/79  14.16.43 ..
000005  10252324140522553555 06241655555555375741 37363655554137413755 55115555555555555555  .HUSTLER 2 FTN     4.6433  6464   I          .
000011  17202454345524220133 05555555555555555555 00000000000000000000 00000000000000000000  .OPT=1 TRACE                                 .
000015  00000000000000000000 00000000000000000000 00000000000000000000 70000000300000000000  .                                  #  C      .
000021  00100002000000000000 06172224120116000000 23312311170000000000 34000001000000000000  . H B        FORTRAN    SYSIO      1  A      .
000025  05300115201405004213 36000006000000000000 05300115201405000000 00000000000000104134  .EXAMPLE 7K3  F         EXAMPLE         A 61.
000031  11162025246000000000 00000000000000100000 17252420252460000000 00000000000001002054  .INPLT#             A   OUTPUT#         A P=.
000035  40000011000001000000 00400002000000030000 00000000000000000000 00000000000000000034  .5   I  A      5 B                         1.
000041  00000000406000000000 00000000000201300000 00000000000000000000 00000000000000000051  .     5#            PC                      (.
000045  00000000000000000000 00000000000000000000 43000002000000000001 00000000000001000007  .                       8  B      A      A  G.
000051  00004100000000000000 40000011000001002054 00400002000000000000 00000000000000000000  .   6          5   I   A P=  5  B            .
000055  00000000000000002110 00000001406000000000 00000000000200300000 00000000000000000000  .          QH      5#            PC          .
000061  00000000000000002125 00000000000000000000 00000000000000000000 43000002000000000001  .          QU                      8  B     A.
000065  00000000000001002063 00004100000000000000 40000020000001004155 10420000004210400000  .        A  P:   6         5  P   A  6 H7   7H5 .
000071  00000000000000000000 00000000000000004162 00000000000000004143 77777777777777777776  .                    6)          68;;;;;;;;;;^.
000075  00000000000000000000 55343333555500000000 51063433573552555555 00000000000000002054  .             100       (F10.2)            P=.
000101  00000000000000004171 00030000000010004177 00030000000100004200 00000000000000000000  .          67 C    A   6; C    A  7          .
000105  55353333555500006000 51565534105535063433 57355255555555555555 40000004000001004174  . 200          (, 1H 2F10.2)      5   O   A 6@.
000111  00000000000000000010 17235000000000000000 17146314631463146315 00000000000000000000  .            OS/          OL:L:L:L:M         .
000115  40000020000001004134 40124212125001125110 51100041300100000000 71700000014360046000  .5   P   A  615J7JJ/  J(H(H  6XA    ?'  A8#0# .
000121  51700041765160004177 51500041767100000003 72750042002006012607 51600041605110004155  .(' 6^(#  6;(/  6^?    C<\  7 PFAVG(# 6#(H 6 .
000125  01000000000007004133 51500041765140004177 52350042003003424700 71000000013665054740  .A        G  60(/  6^(5  6;)2  7 XC7* ?   A3_E*5.
000131  54650710000001227706 03370041100400004137 51500041775140004175 40745517000420046000  .= GH  AS;FC4  6/D   64(/  6;(5  6\5@ O  OPD# .
000135  51100041644600046000 40000005000001004130 10401000000000000000 11162025240000000000  .(F 6"-  O# 5   E   A  6XH5H        INPUT     .
000141  17252420252400002034 77777777777777636257 05300115201405004134 40000003000001004153  .OUTPUT   P=;;;;;;;;:].EXAMFLE 615   C   A 6$.
000145  12000000000000000000 01000000000016004133 51100041330400000000 44000006000000000000  .J              A        N 60(H 60D     9  F   .
000151  05160457000000000000 40010041141725240311 57000000000600104153 11162003115700000000  .END.          5A  6=OUTCI.      #A 6$INPCI.  .
000155  60010041422134162422 31570000004001004134 46000001000000000000 05300115201405000000  .#A 67Q1NTRY.     5A  61-   A          EXAMPLE   .

EOR OF LEVEL 000 FOUND,  BINARY RECORC HAS    1120   0001608 WORDS.
DUMPING TERMINATED AT LEVEL 008.        1 RECORDS DUMPED.
20.53.22..000004 PAGES PRINT. 000057 LINES PRINT. FOR $ 000.07 AT RC2.
```

Figure 7.2: FILEDMP Output (Example 2)

In this example, the first section of file TAPE1 is read in coded mode and dumped to file OUTPUT in Hollerith format. The next section is skipped, and the third section is read in binary mode and dumped in octal and Hollerith. Figure 7.2 shows the output from this example.

7.6.3 LISTTY

LISTTY is a utility for listing a coded file. LISTTY is called as follows:

```
LISTTY[,optional parameters].
```

If run from batch and no parameters are given, LISTTY will read from file FILE and list the entire contents on OUTPUT. A carriage control '1' precedes the first listed line, causing the listing to start at the top of a new page. If run at an interactive terminal, LISTTY will list FILE on TTYTTY and will output only columns 1-72 of each line. Each line will be prefixed by a blank (the carriage control), a four-digit line number generated by LISTTY, and a single character which is blank for lines read from the file or "*" indicating EOS, EOP, or EOI. (If a file contains more than 9999 lines, the last four digits of the line number are used by LISTTY.) Normally LISTTY will list only the first line of a group of identical lines, and will prefix the next line listed with an equal sign to indicate that lines were skipped. The input file is rewound before listing unless the NR parameter is specified.

The parameters for LISTTY may appear in any order. Continuation cards may be used, but no parameter may be split between two cards since there is an implied delimiter between successive cards. Legal delimiters are the comma and left parenthesis; terminators are the period and right parenthesis. The following parameters are recognized.

ALL list all lines selected by the other options. This option suppresses the skipping of duplicate lines after the first has been printed.

B suppress printing of extra blanks. Occurrences of two or more consecutive blank characters are reduced to a single blank. This option also selects a default line width of 137.

CCx use 'x' as the carriage control for each line (e.g., CC0 for double space). The default carriage control is a blank.

CLEAR return file ZZZZDFS, which clears the saved parameters before the other parameters of the current statement are processed.

Cn list lines starting at column n. The default is 1; maximum is 137.

Cn-m same as Cn,Wm.

COPY suppress the addition of the carriage control and the printing of the line numbers. This option also selects ALL.

HELP print a full HELP listing, explaining how to use LISTTY.

I=inlfn specifies the name of the input file. The default input file is FILE.

ID=/chars/ LISTTY begins the listing at the first page whose top line contains the specified string. A top-of-page line is identified by a "1" or "T" carriage control. This option is useful for locating a program listing within the input file.

Lm stop listing the file at line m. The default is 0, which means listing continues until end-of-information. This parameter may appear up to 20 times, in ascending order, each occurrence following the corresponding Sn parameter.

n-m same as Sn,Lm.

n-m/chars/[N][U]

This parameter specifies a character string search and consists of four components: the string (delimited by slash marks), an optional column number (n) or column range (n-m), a unit option (U), and a no-match option (N).

LISTTY will list only the lines that contain the specified string; or, if N is used, only the lines that do not contain the string. If U is declared, the string must appear between non-alphanumeric characters. If a column number or column range is specified, the string must start within the indicated column(s).

The string may be up to 50 characters. A slash within the string must be represented by two consecutive slashes (//). A period, right parenthesis or a dollar sign must be represented by a slash followed by the octal Display code (i.e. /57, /52, and /53, respectively) in interactive usage; the dollar sign must appear as /53 in batch usage as well. Other characters may also be represented in this manner.

NR list the input file from its present position (i.e., no rewind).

NS omit line numbers in the output.

O=outlfn specifies the output file name. From batch, the default is OUTPUT. From interactive, the default is TTYTTY, which means that the listing is delayed until LISTTY has terminated; if the keyword O appears alone, output will be on the connected file ZZZZOT, which causes immediate output to be terminal.Note: If O=outlfn is specificed, the default line width is 137 characters rather than 72.

If this parameter is allowed to default in an interactive job, then LISTTY will list the first 66 characters of a line, then continue with the 67th character on a new output line; the continuation line is not indented.

Pn-m specifies a page number (Pn) or page range (Pn-m) to be listed. This parameter may appear up to 10 times. A page number is assumed to be the rightmost numeric field in a top-of-page line. **Caution:** Using the S and P options simultaneously will produce unpredictable results if the starting line number is greater than the line numbers of the first page or page range.

SAVE requests that the parameters of this LISTTY call be saved on file ZZZZDFS. Subsequent calls to LISTTY will be executed as if these parameters were included among those appearing in the later calls. This option suppresses the listing; that is, only the SAVE operation is performed.

Sn lists the file starting at line n. The default is 1. This parameter may appear up to 20 times in one LISTTY statement, in ascending order.

Wm lists lines up to and including column m. The default column width is normally 72 for interactive jobs, but is changed to 137 if the B or O=outlfn parameter is selected. The default for batch jobs is always 137 which is also the maximum.

Z suppresses printing of the end-of-sections.

Example:

```
PNC
job card
PW=password
ATTACH,LISTING,FORTRANPROGRAMPGM.
LISTTY,I=LISTING,COPY,P1-2,ID=/PGM/.
6/7/8/9
```

This job attaches a FORTRAN program named PGM, copies pages 1 and 2 of program PGM from file LISTING to file OUTPUT, omitting the carriage control character and sequence numbers that are normally generated by LISTTY.

7.6.4 PRINTLB

PRINTLB prints the contents of the label of a magnetic tape assigned to a user's job.

```
PRINTLB,[inlfn],[outlfn].
```

The parameters are optional but order-dependent.

inlfn the local file name assigned to the tape by the REQUEST statement. The default is TAPE1.

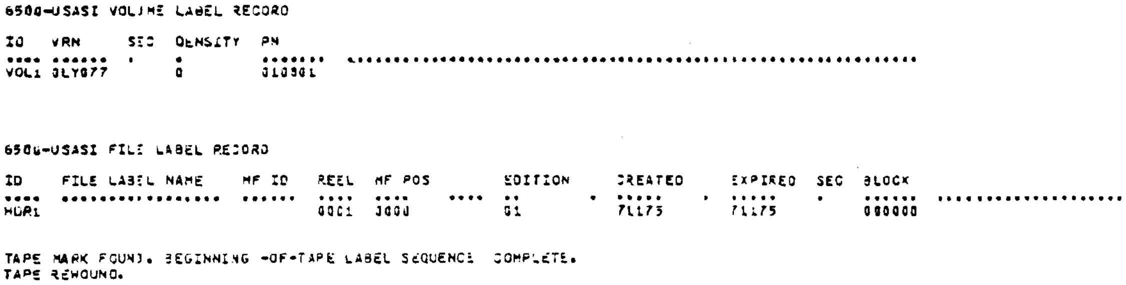
outlfn the local file where PRINTLB is to write its output. The default file name is OUTPUT.

The S (stranger), Z (unlabeled) and NS (nonstandard label processing) parameters must be specified on the REQUEST statement for the tape. PRINTLB reads the first blocks and, if it finds a recognizable 6500 or 3600 label, copies the label to file outlfn.

Example:

```
PNC
id,MT1.
PW=password
REQUEST,TAPE1,VRN=LY077,Z,S,NS.
PRINTLB,TAPE1.
6/7/8/9
```

In this example, the label for tape LY077 is displayed on the file OUTPUT. The output is shown in Figure 7.3.



```
6500-USASI VOLUME LABEL RECORD

ID   VRN     SEC  DENSITY  PN
•••• ••••••  •    •        •••••••  •••••••••••••••••••••••••••••••••••••••••••••••••••••••••••••
VOL1 0LY077       0        010301


6500-USASI FILE LABEL RECORD

ID    FILE LABEL NAME    MF ID   REEL  MF POS        EDITION      CREATED     EXPIRED  SEC  BLOCK
••••  •••••••••••••••••  ••••••  ••••  ••••     •••• ••         • •••••     • •••••    •    ••••••  •••••••••••••••••••
HDR1                             0001  0000          01           71175       71175         000000

TAPE MARK FOUND. BEGINNING -OF-TAPE LABEL SEQUENCE  COMPLETE.
TAPE REWOUND.
```

Figure 7.3: PRINTLB Output

7.6.5 SCAN

The SCAN control statement prints a summary of the contents of a binary file of programs.

SCAN,[binlfn],[outlfn].

The parameters are optional but order-dependent.

binlfn the local file name of a file containing relocatable object modules and/or absolute overlays. The default is TAPE1.

outlfn the local file name where SCAN is to write its output. The default is OUTPUT (connected in an interactive job).

SCAN prints one line for each SCOPE section in binlfn. The information in a line depends on the contents of the section:

Type	*Information*
Relocatable module	name and length
Absolute overlay	name, level numbers, load address and entry point address
OVERLAY directive	the directive is printed
Empty section	SCOPE end-of-section level
other	SCAN prints "UNRECOG.—STARTS" followed by the first four words of the section in octal.

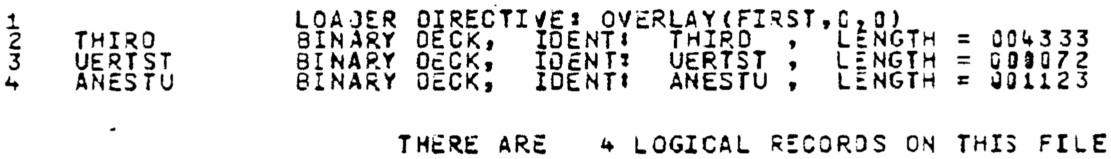
At the end, SCAN prints the number of sections that it reads. SCAN will stop reading binlfn when it encounters an end-of-partition. SCAN does not rewind either file binlfn or file outlfn either before or after the scan.

Example:

```
PNC
job card
PW=password
FTN.
HAL,LGO=subroutines.
LGO.
REWIND,LGO.
SCAN,LGO.
7/8/9
          FORTRAN program
6/7/8/9
```

This job compiles a FORTRAN program and writes the object code on the file LGO, writes FORTRAN subroutines from the HUSTLER Auxiliary Library onto LGO, and executes the program; SCAN is then used to print a summary of the contents of LGO.

Figure 7.4 shows the output from this example.



```
1                         LOADER DIRECTIVE: OVERLAY(FIRST,0,0)
2   THIRD                 BINARY DECK,  IDENT:  THIRD  ,  LENGTH = 004333
3   UERTST                BINARY DECK,  IDENT:  UERTST ,  LENGTH = 000072
4   ANESTU                BINARY DECK,  IDENT:  ANESTU ,  LENGTH = 001123

                              THERE ARE   4 LOGICAL RECORDS ON THIS FILE
```

Figure 7.4: SCAN Output

7.7 File Manipulation

Control statements in this category allow the user to control file positioning.

7.7.1 BKSP

BKSP backspaces a file a specified number of sections.

```
BKSP,lfn[,n].
```

lfn local file name of the file to be backspaced. If lfn does not exist, it is created as an empty file. If lfn is a tape, it must be a coded tape.

If lfn is positioned within a section, that section is counted as the first one backspaced over.

n optional number (decimal) of sections to be backspaced over. The default is 1. The maximum number that may be specified is 262143_{10}.

If n is an illegal value, the job will abort.

Backspacing stops if beginning-of-information is encountered.

Note: BKSP will not backspace past the beginning of a tape volume. BKSP cannot be used on binary SCOPE 9-track tapes.

Example:

```
PNC
job card
PW=password
ATTACH,X,DMSPERMFILE.
COPYSBF,X,OUTPUT.
BKSP,X,2.
COPY,X,PUNCHC.
6/7/8/9
```

This job copies an entire file to OUTPUT, then backspaces two sections and copies those two sections to PUNCHC.

7.7.2 REWIND

The purpose of the REWIND control statement is to position a file at the beginning-of-information.

```
REWIND,lfn[,...].
```

lfn local file name(s) of the file(s) to be rewound. If lfn does not exist, it is created on disk.

On a multivolume magnetic tape file, REWIND will back up to the beginning of the first reel.

Example:

```
PNC
job card
PW=password
ATTACH,X,DATAFILE.
COPY,X,PUNCHC.
REWIND,X.
COPYSBF,X,OUTPUT.
6/7/8/9
```

This job causes file X to be copied to PUNCHC (output on punched cards), then rewound and copied to OUTPUT (printed output).

7.7.3 SKIPB and SKIPF

The SKIPF and SKIPB control statements reposition a file by skipping forward (SKIPF) or backward (SKIPB).

```
SKIPF,[lfn],[n],[lev],[mod].

SKIPB,[lfn],[n],[lev],[mod].
```

The parameters are optional, but position-dependent; a parameter that is omitted must be replaced by a null parameter (see examples).

lfn the local file name of the file to be repositioned. The default name is FILE.

n decimal number of SCOPE sections to be skipped ($1 \leq n \leq 262143$). The default is 1.

If n=262143, SKIPB performs a rewind; SKIPF skips to end-of-information (EOI) on a disk or 9-track tape file, but does not reposition a 7-track tape file. Using 262143 for n is recommended when performing a SKIPF on a disk file, since it does not read the disk file and costs less.

lev section level number in octal ($0 \leq lev \leq 17$). Only sections of level lev or above are included in the skip count. The default is 0. Level 17 is equivalent to EOP.

mod either of the following modes:

B binary (this is the default).

C coded (must be specified for coded (even-parity) 7-track tapes and coded SCOPE 9-track tapes).

Any illegal parameters will cause a CPU abort.

SKIPF and SKIPB count sections by counting end-of-section marks, leaving the file positioned just after the last end-of-section that was counted. If a file is initially positioned just after an end-of-section, that end-of-section is not included in the skip count.

Examples:

```
SKIPF.
```

This is equivalent to:

```
SKIPF,FILE,1,0,B.
```

```
SKIPB,,,17.
```

This is equivalent to:

```
SKIPB,FILE,1,17,B.
```

7.8 File Editing

This section describes a utility that has the ability to modify a file line by line.

7.8.1 EDITOR

EDITOR is the name of the text editing system that enables the interactive user to build and edit files from an interactive terminal. EDITOR can also be used from batch by executing the control statement shown below. For those who are unfamiliar with EDITOR, a short description of its capabilities is included in this section. For a full description see Chapter 3 in the *Interactive System User's Guide*.

```
EDITOR[,I=inlfn][,L=listlfn][,E=ewflfn].
```

I=inlfn specifies the name of the file from which EDITOR directives and numbered text lines will be read. The default file name is INPUT.

L=listlfn specifies the name of the file onto which diagnostics and work file listings will be written. The default file name is OUTPUT.

E=ewflfn specifies the name of the EDITOR work file. This file must already be in the work file format, i.e., it must have been previously created by EDITOR. If omitted, EDITOR creates an empty work file named EWFILE. Alternate methods of specifying the work file are illustrated in the examples at the end of this section.

The input file (I=inlfn) from which EDITOR reads should contain card images of directives and text lines in exactly the same format as they would be typed at a terminal; each card image corresponds to a line of terminal input. The input file must not contain SCOPE/HUSTLER control statements or special interactive commands (including N, the auto-numbering command) or the job will abort. Any EDITOR operation that can be done interactively can also be done through batch, subject to the following cautions.

1. A 'Y' (YES) will automatically be entered on any operation that employs the veto option.
2. If a compilation directive (e.g., FTN, GO, BASIC, etc.) is issued, all control statements following the EDITOR statement will be skipped.
3. The LIST directive will by default use the FULL option (no suppression of blanks).
4. The USE directive instructs EDITOR to work with a different work file, but the file will not be renamed EWFILE as in interactive mode.

The Work File

Note: The following overview of the EDITOR system was written with the batch user in mind.

All EDITOR directives operate on the contents of a random-access, compressed-format disk file, known as the EDITOR work file. In interactive mode this file is always named EWFILE. In batch mode, the work file is specified by the E=ewflfn parameter or by a USE directive, but in neither case is the specified file renamed.

The work file is composed of text lines, each of which is associated with a unique line number ranging from 0.000001 to 999999.999999. The text lines are automatically arranged in order of increasing line number. A particular line may be replaced by entering a new text line with the same line number. When the SAVE directive is used to copy text lines from the work file to a standard SCOPE file, each text line is converted to a record.

Like any other disk file, the EDITOR work file may be cataloged as a permanent file. Once a work file has been cataloged, all subsequent modifications will automatically be made permanent provided that the file is attached with read, modify, and extend permissions. After attaching a previously created work file, it may be declared as the current work file by either the E=ewflfn parameter or the USE directive.

Creating a Work File

The user may enter text lines into the work file in the following ways:

1. The OLD directive instructs EDITOR to read in lines from a standard coded file, assigning line numbers sequentially or using numbers already contained in the lines. The work file must be empty when OLD is issued.

2. Text line cards may be interspersed with directives on the file specified by I=inlfn. A text line card is one which begins with an EDITOR line number.

3. Text line cards may be interspersed with directives on the file specified in a READ directive. READ should be the last directive in the input file, since it causes reading to cease on the input file and commence on the file specified by the READ.

4. The MERGE directive instructs EDITOR to merge numbered lines from a standard coded file with the contents of the work file.

5. The INSERT directive instructs EDITOR to insert lines from a standard coded file between two existing lines of the work file.

Editing Features

In addition to the MERGE and INSERT directives, EDITOR provides the following features for manipulating the contents of the work file.

1. The RESEQ directive renumbers the text lines without altering their sequence.

2. The DUP directive duplicates specified text lines at other locations of the work file.

3. The MOVE directive relocates specified lines within the work file, i.e., assigns them new line numbers.

4. The DELETE directive is used to delete a set of text lines. An individual line can also be deleted by means of a text line card which contains only the line number.

5. A set of intra-line editing directives allows the user to insert, delete, or replace specified text within an existing text line or set of text lines.

Most of these directives can be applied to a set of text lines defined by (1) line number(s) or line number range(s), (2) the presence or absence of a particular string of characters within the lines, or (3) a combination of 1 and 2. This feature also applies to the output directives described below.

Output Operations

Because of the special format of the work file, only EDITOR can successfully use the file's contents. As previously remarked, the user can convert and copy selected text lines to a standard coded file by using the SAVE directive. He/she can also direct EDITOR to compile and execute a program within the work file. The LIST directive is used to list selected text lines on the file specified by L=listlfn (or the default file OUTPUT), with or without associated line numbers.

Example 1: Using OLD directive to enter unnumbered lines.

```
PNC
job card
PW=password
COPYCR,INPUT,A.
EDITOR.
CATALOG,EWFILE,MYWORKFILE.
7/8/9
text cards (unnumbered)
7/8/9
LENGTH,80
OLD,A FROM 100 BY 10.
LIST.
6/7/8/9
```

This job enters lines from file A (after copying them from INPUT) and assigns them line numbers in increments of 10, starting from 100.

Example 2: Adding text to a previously created work file.

```
PNC
job card
PW=password
COPYCR,INPUT,A.
ATTACH,OLDFILE,MYWORKFILE.
EDITOR,E=OLDFILE.
7/8/9
text cards
7/8/9
INSERT A AT *L
LIST.
6/7/8/9
```

This job copies a set of text cards to file A and then calls EDITOR to add those cards to the end of the work file (the permanent file MYWORKFILE).

Example 3: The USE directive.

```
PNC
job card
PW=password
ATTACH,A,EWFILE1.
ATTACH,B,EWFILE2.
EDITOR,E=B.
7/8/9
SAVE,TEMP.
SCRATCH.
USE,A.
MERGE,TEMP,FROM,105,BY,5.
LIST.
6/7/8/9
```

This job merges two work files. First the contents of EWFILE2 (local file B) are saved on TEMP. Next B is released and EWFILE1 (local file A) is declared as the work file. Lastly, TEMP is merged with the work file and the resulting contents are listed.

Example 4: Intra-line editing.

```
PNC
job card
PW=password
ATTACH,XXX,MYEWFILE.
EDITOR,E=XXX.
7/8/9
/VALUE/U=/IVALUE/,LIST,ALL
FTNER
6/7/8/9
```

This job illustrates the value of intra-line editing. With a single directive, the user changes the variable name VALUE to IVALUE. He/she also requests a list of all lines modified. The FTNER directive causes the text to be saved and compiled by FTN; then the compiled program is loaded and executed. If errors occur during compilation, ERRS is called, and error messages will be written to the file OUTPUT.

7.9 Loader Control

The loader is the system program that places object code into central memory and makes it ready for execution. It may also perform related services such as printing a load map or presetting initialized memory.

A load sequence (also called a load set or load operation) encompasses all of the loader's processing from the time that the loader is called until the time the loaded program is ready to execute. The load sequence must not be interrupted by statements unrelated to loading, such as file positioning commands (e.g. REWIND) or requests for system resources (e.g. REQUEST). A load sequence is terminated by one of the following: a NOGO statement, an EXECUTE statement, or a control statement which is not a loader control statement (such as LGO). The loader control statements are:

```
EXECUTE     NOGO
LDSET       SATISFY
LIBLOAD     SEGLOAD
LOAD        SLOAD
name
```

(DMP, MAP, and REDUCE are allowed within a load sequence for compatibility with previous versions of the loader.) In particular, the following are not loader control statements; if they appear within a load sequence, the job will usually abort: AUTORFL, LIBRARY, MFL, and RFL.

The loader is documented in the CDC *Cyber Loader Reference Manual* (Publication No. 60344200); further documentation may be found in the *User's Guide Supplement: LIBEDIT—Cyber Loader Libraries.*

7.9.1 EXECUTE

The EXECUTE statement terminates a load sequence and initiates execution of the loaded program.

```
EXECUTE,[ept],[exparam[,...]].
```

The parameters are optional but position-dependent; an omitted parameter must be replaced by a null parameter.

ept — name of an alternate entry point at which execution is to begin.

By default, execution begins at either (1) the last transfer symbol (main program) encountered in a relocatable load; or (2) the first entry point encountered in a core image (overlay) load.

If ept is omitted and no default is found, a fatal error results. If ept is specified but does not exist, the default is used with no error.

exparam — execution-time parameter(s) to be passed to the loaded program.

Examples: Full documentation and examples are available in the *Cyber Loader Reference Manual* (Publication No. 60344200).

7.9.2 LDSET

LDSET controls a variety of load options. **It applies only to the current load sequence.**

LDSET has the form:

LDSET,option(s).

The options consist of keywords. Legal keywords are:

EPT
ERR
LIB
MAP
HEXMAP
NOEPT
NOREWIN
OMIT
PRESET
PRESETA
REWIND
STAT
SUBST
USE
USEP

Keywords may appear alone or may be followed by one or more values. Thus, two forms are allowed:

keyword
keyword=value[/value...]

Note that if more than one value for a keyword is specified, the values are separated by slashes (/). Each keyword is discussed separately in the following pages.

In general, LDSET options take effect as they are encountered and remain in effect until the occurrence of a countermanding LDSET or the end of the load sequence, whichever comes first. Exceptions to this are noted in the individual descriptions.

Options may be listed on one or more LDSET statements; their effect is the same.

LDSET,EPT=ept_1/.../ept_n.
LDSET,NOEPT=ept_1/.../ept_n.
LDSET,NOEPT.

This statement provides control over the entry points of capsules, overlays and OVCAPs (overlay-capsules).

Capsules are linked to each other and to the calling program through the use of entry points and external references. An external reference generated by one of the programs comprising the capsule becomes an external reference of the capsule if it is not satisfied by an entry point of one of the other programs in the capsule.

The entry points of a capsule are normally those that are defined by programs within the capsule and are not referenced within the capsule. This set can be altered through the following LDSET options:

EPT=ept_i specifies one or more entry points that are to appear as entry points of the capsule, regardless of whether the entry points are referenced within the capsule.

NOEPT=ept_i prevents the listed entry points from becoming entry points of the capsule even if they are not referenced within the capsule.

NOEPT specifies that the only entry points are to be those explicitly mentioned by the EPT request and an entry point whose name is the same as the capsule name (if any).

Examples: See the CDC *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2, for examples.

LDSET,ERR=errlev.

This statement determines what level of loader errors will cause the job to abort. This statement applies only to the current load sequence. The following words may be used for errlev (only one may be selected):

ALL abort for fatal and non-fatal errors.

FATAL abort for fatal errors. This is the default.

NONE if possible, continue processing even after fatal errors.

Examples: Examples may be found in the *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2.

LDSET,LIB[=libname].

This statement adds the named libraries to the end of the local library set, or it can be used to clear the local library set. This applies only to the current load sequence.

The global library set is defined by the LIBRARY control statement (see Section 7.10.1).

libname name of a system library or a local file in library format. If a local file has the same name as a system library, the local file is used, except that NUCLEUS always refers to the system library. More than one libname may be specified, separated by slashes (/). LIB with no libnames clears the local library set.

The file libname is ignored if it is already in the global or local library set.

The libraries are added to the local set in the sequence specified.

Note: Programs that require the use of 8-bit subroutines use a 'LDSET,LIB=BIT8LIB.' statement for execution.

Examples: Examples may be found in the *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2.

LDSET,{MAP|HEXMAP}=[maptype][/maplfn].

This statement determines the contents of the load map and the file on which it is written. (HEXMAP produces a hexadecimal map.) This applies only to the current load sequence.

maptype may be the letter N or any combination of the letters S, B, X, or E:

type	map contents
N	no map
S	statistics only
B	block names and addresses only
X	external/entry point cross-reference only
E	entry point names and addresses only

The default map type for the job may be set by the MAP control statement (see Section 7.10.2), but may be overridden in the current load sequence by S, B, X, or E.

If type is omitted, map contents are determined by the last MAP control statement, or, if none, by the job default (SB in a batch job, N in an interactive job).

maplfn the file that contains the map contents. The file is not rewound. If omitted, the default is OUTPUT in a batch run, MAPFILE in an interactive job.

Note: A map is never written for an error-free absolute core-image load.

Examples: Examples may be found in the *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2.

LDSET,OMIT=ept[/...].

This statement inhibits the loader from satisfying external references to the named entry point(s). This applies only to the current load sequence.

OMIT requests are cumulative, i.e., a subsequent OMIT does not cancel the effect of a previous OMIT.

ept entry point name(s). References in other modules to ept will not be satisfied, even if modules containing the entry points are loaded for other reasons. The specified entry point names are processed the same as other unsatisfied names but do not result in errors.

The ept should not appear in an 'LDSET,SUBST.' statement (see below) in the same load; results are unpredictable.

Examples: Examples may be found in the *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2.

LDSET,PRESET[A]=[val].

This statement specifies a value to be stored in memory locations whose contents are otherwise undefined. It applies only to the current load sequence.

'LDSET,PRESET.' and 'LDSET,PRESETA.' statements take effect as they are encountered.

val may be either of the following:

1. a 1-20 digit octal number, optionally prefixed by + or -, and optionally suffixed by a B.

2. one of the following keywords that represent a value:

Keyword	Octal Preset Value				
NONE	no preset				
ZERO	0000	0000	0000	0000	0000
ONES	7777	7777	7777	7777	7777
INDEF	1777	0000	0000	0000	0000
INF	3777	0000	0000	0000	0000
NGINDEF	6000	0000	0000	0000	0000
NGINF	4000	0000	0000	0000	0000
ALTZERO	2525	2525	2525	2525	2525
ALTONES	5252	5252	5252	5252	5252
DEBUG	6000	0000	0004	0040	0000

PRESETA=DEBUG is the default.

If PRESETA is used instead of PRESET, the result is the same, except that the low order 17 bits of each word will contain the word's address. (Exception: PRESETA=NONE means no preset).

Examples: Examples may be found in the *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2.

LDSET{,REWIND|,NOREWIN}.

This statement establishes the default rewind indicator for subsequent LOAD, SLOAD, and 'name' call statements. Rewind indicators /R and /NR on LOAD and SLOAD statements override the defaults established by this statement. It applies only to the current load sequence.

REWIND All files except INPUT will be rewound before processing by LOAD, SLOAD, or 'name' call statements. This is the default.

NOREWIN Load files will not be rewound before processing. (Note the spelling of this keyword; 'NOREWIND' is **illegal**.)

Note: Load files are never rewound after loading.

Examples: Examples may be found in the *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2.

LDSET,STAT=lfn[/...].

This statement instructs the loader to load relocatable Cyber Record Manager modules to process the specified local file(s). The system library CRM must be in the job's library set; otherwise a non-fatal "UNSATISFIED EXTERNAL" error occurs.

lfn name of a local file(s) described in a previous FILE statement (see Section 7.4.2). If more than one file is specified, they must be separated by slashes (/).

Examples:

Examples may be found in the *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2.

LDSET,SUBST=eptold-eptnew[/...].

With this statement, external references to entry point names are changed into references to new entry point names; in other words, external references to eptold are treated as external references to eptnew. This statement applies only to the current load sequence.

Entry point names in SUBST requests should not appear in 'LDSET,OMIT.' statements in the same load sequence; the results are unpredictable.

eptold name of the entry point to which external references originally referred.

eptnew name of the entry point used to satisfy external references to eptold.

Examples:

SUBST requests are not transitive; the example below illustrates this.

```
LDSET,SUBST=A-B/B-C.
```

Original references to A are treated as references to B; original references to B are treated as references to C; but C is not substituted for A.

In the event of a conflict between SUBST requests, the most recent one takes precedence. For example, in the statement

```
LDSET,SUBST=A-B/A-C.
```

original references to A are treated as references to C.

More examples are available in the *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2.

LDSET,USEP=progname[/...].
LDSET,USE=eptname[/...].

'LDSET,USEP.' forces the loading of the named relocatable programs. 'LDSET,USE.' forces loading of relocatable programs containing the named entry points. The requested modules are loaded on the next occasion that the loader satisfies externals.

These statements apply only to the current load sequence. If the load sequence involves OVERLAY directives, the USE or USEP applies only to the most recently generated overlay; or, if it occurred prior to the first LOAD statement, it applies only to the first overlay.

progname program names.

eptname entry point names.

If one or more of progname or eptname is not found in the libraries searched, a non-fatal error results.

Examples: Examples may be found in the *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2.

7.9.3 LIBLOAD

Modules containing the named entry points are loaded from the specified library with the LIBLOAD control statement. The load sequence is not terminated.

```
LIBLOAD,libname,eptname[,...].
```

libname the name of a system library, or a local file in library format. If a local file has the same name as a system library, the local file is used, except that NUCLEUS always refers to the system library. Note that the indicated library need not be in the current global or local library set.

eptname name of an entry point in a module on the named library. Naming a nonexistent entry point causes a non-fatal error.

The entry point names may refer to relocatable or core image modules, but not both.

If more than one entry point name refers to the same module, fewer modules than names are loaded.

In a core image load, only one module may be loaded, and an EXECUTE statement must follow LIBLOAD; otherwise a non-fatal error occurs.

Examples: Examples may be found in the CDC *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2.

7.9.4 LOAD

The LOAD statement causes the loading of programs from the named file(s). It does not terminate a load sequence.

```
LOAD,loadlfn[,...].
```

loadlfn may be one of the these forms:

loadlfn/R Rewind local file loadlfn, then load. This is the default for files other than INPUT.

loadlfn/NR Load local file loadlfn without rewinding. This is the default for the file INPUT.

loadlfn Use default rewind indicator.

A previous 'LDSET,NOREWIN.' statement (see Section 7.9.2) changes the default rewind indicator to NR for all files.

In a relocatable load, files are loaded in the order specified. Programs are loaded from a file until end-of-information, end-of-partition, or double end-of-section is encountered.

Example 1: LGO1 and LGO2 contain relocatable modules.

```
LOAD,INPUT.             INPUT is not rewound before loading.
LOAD,LGO1,LGO2/NR.      LGO1 is rewound, then loaded. LGO2 is not rewound
                        before loading.
NOGO.                   Terminates load sequence.
```

In a core image load, only one file may be specified and only one module (which must be level 0,0) is loaded. An EXECUTE statement must follow.

Example 2: File ABS is positioned at the beginning of a (0,0) overlay.

```
LDSET,NOREWIN.          Change default to NR.
LOAD,ABS.               The (0,0) overlay is loaded.
EXECUTE.                Required.
```

Additional examples may be found in the CDC *Cyber Loader Reference Manual* (Publication No. 60344200), Section 2.

7.9.5 name

The 'name' call statement is used to load and execute programs on local files or in libraries; 'name' is the local file or library name. It always terminates a load sequence.

```
[+]name[,exparams].
```

+ instructs the loader to start its search for name in the library NUCLEUS (see E below).

exparams optional execution-time parameters to be passed to the loaded program. The exact form depends on the program being loaded.

The system loads and executes whichever of the following is found first. (Note: Steps A and C are omitted if the 'name' call occurs within a load sequence.)

A. Special cases

COMMENT, EXEC, EXIT, MODE and RFL always invoke the corresponding system program. In addition, for an interactive job, the special case list includes all interactive and EDITOR commands (see the *Interactive System User's Guide*).

B. Local file

Programs on local file 'name' are loaded and executed. By default, the file is rewound first (except INPUT). This may be altered by the use of the 'LDSET,NOREWIN.' statement. If the file contains core image modules (e.g. overlays), the first one is loaded and execution begins at the first or only entry point listed in the header of the module. If the file contains relocatable modules, programs are loaded until one of the following is encountered: end-of-information, end-of-partition, or double end-of-section. Execution then begins at the last transfer symbol (main program) loaded.

If the file name call is part of an overlay-generation or segment-generation load sequence the following occurs: the core-image modules (overlays or segments) are generated and written to the specified file; the main overlay or root segment is loaded; and execution begins at the first entry point in the header of the loaded module. Note that the first overlay on the file is executed, even if the load generates a second main overlay.

C. Control Statement-callable peripheral processor programs.

Currently this includes only DMP, MFL, and RTL.

D. Library program containing entry point 'name'.

The program is loaded from the library, externals are satisfied for a relocatable program, and execution begins at the entry point 'name'.

Libraries are searched in the following order:

1. Global library set—defined by a LIBRARY statement.

2. Local library set—defined by 'LDSET,LIB=' statements.

E. NUCLEUS Library

The program 'name' or the program containing entry point 'name' is loaded. NUCLEUS is where most of the programs "on the system" reside. If the search started here due to a '+' prefix and 'name' is not found, the search proceeds to steps B, C and D.

Examples: See the *Cyber Loader Reference Manual* (Publication No. 60344200), Section 2, for examples.

7.9.6 NOGO

NOGO terminates an object module load sequence without beginning execution of the loaded program. Optionally, the core image of the loaded program may be written on a local file as a core-image module; entry point names to be included in the header of this core image module may also be specified.

The use of NOGO to terminate a core-image load sequence results in a non-fatal error and all parameters are ignored.

```
NOGO[,outlfn][,eptname][,...].
```

outlfn — the name of the output file on which the core image is to be written. The output file can be specified either on the NOGO statement or on overlay directives, if present. If both are present, the file on the NOGO statement will be used. If outlfn is not given with NOGO, the file name(s) on the OVERLAY directive(s) will be used. If no file is given on either statement, the core image will not be written.

eptname — optional entry point names to be made available to EXECUTE statements when the core image is loaded and executed. If eptname is omitted, the default entry point is the last transfer symbol (main program). If the load included overlay directives, eptname is ignored and the entry point list in the core image module includes only the default entry point.

Examples: Examples may be found in the CDC *Cyber Loader Reference Manual* (Publication No. 60344200), Section 2.

7.9.7 SATISFY

SATISFY causes the loader to satisfy external references immediately instead of waiting until load completion; it optionally specifies the libraries to be used.

```
SATISFY,[libname][,...].
```

libname — the name of a system library, or a name of a file in Cyber Loader library format. Libraries are searched in the order specified. If no libnames are specified, the currently defined library set (global set, local set, NUCLEUS) is used. Naming a nonexistent or ill-formatted library causes a non-fatal error; the SATISFY statement is then processed using any remaining libnames. If a local file has the same name as a system library, the local file is used (with the exception that NUCLEUS always refers to the system library).

SATISFY does not alter the library set.

If the load includes the use of OVERLAY statements, then SATISFY applies only to the last overlay generated before the SATISFY statement.

Examples: See the CDC *Cyber Loader Reference Manual* (Publication No. 60344200), Section 2, for examples.

7.9.8 SEGLOAD

SEGLOAD initiates segment generation. It must be the first statement in the load sequence.

```
SEGLOAD[,I=dirlfn][,B=abslfn][,LO=c₁c₂].
```

I=dirlfn the name of the local file from which segmentation directives are read. The default is INPUT.

B=abslfn the name of the local file onto which segmented binary output is written. The default is ABS.

LO=c_1c_2 list options for the segment loader. The characters c_1 and c_2 control the directive list and the tree diagram. c_1c_2 values are:

0	Neither tree diagram nor directive list.
D	Directive list only.
T	Tree diagram only.
DT	Both directive
TD	list and tree
omitted	diagram

If the LO parameter is omitted, the list options depend on the load map produced. If MAP(OFF) or LDSET(MAP=N) is specified, LO=0 is assumed; otherwise, LO=DT is assumed.

Examples: Examples of SEGLOAD may be found in the *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2. Segmentation is discussed in full in Chapter 5 of that manual.

7.9.9 SLOAD

SLOAD requests the loader to load the named programs from the file specified.

```
SLOAD,lfn[/R|/NR],progname[,...].
```

lfn the local file from which programs are to be loaded. File lfn must be sequential. This parameter is required. A fatal error occurs if lfn does not exist.

/R optional suffix to lfn. Rewind the file before searching for programs. This is the default for all files but INPUT.

/NR optional suffix to lfn. Do not rewind the file before searching. This is the default for file INPUT.

progname the name(s) of the program(s) to be loaded. A non-fatal error occurs if progname is not found.

The default rewind indicator may be changed to NR by the 'LDSET,NOREWIN.' statement (see Section 7.9.2).

Programs are loaded in the order in which they are found on lfn. The file is searched from its current position (after rewinding if appropriate) until either: (1) all specified programs are loaded; or (2) an end-of-information, end-of-partition, or double end-of-section is encountered. The file is then left in its current position. SLOAD does not begin execution of the program.

When loading a core image module, only one name may be specified; otherwise, a fatal error occurs. A non-fatal error occurs if an EXECUTE statement does not follow the load of a core-image module.

Examples: Examples may be found in the CDC *Cyber Loader Reference Manual* (Publication No. 60429800), Section 2.

7.10 Loader-Related Control Statements

The following statements have functions related to the loading process, but are not part of the load sequence.

7.10.1 LIBRARY

The LIBRARY statement defines the global library set. It may not appear within a load sequence.

```
LIBRARY[,libname][,...].
```

libname   name of a system library or local file in library format. If a local file has the same name as a system library, the local file is used, except that NUCLEUS always refers to the system library. If no libnames are specified, the global library set is empty (this is the default condition).

The maximum number of libnames allowed depends on how many of the libnames do not match the names of existing system libraries. (In this discussion a "system name" refers to a name that matches an existing system library; a name that is not a "system name" is a "user name.") The following table shows how the number of libnames varies with the number of "user names."

number of user names	max. number of libnames on LIBRARY statement
0	24
1	14
2	4

As you can see, this scheme does not allow for the use of many user libraries unless some of the local files are given "system names." Noting this restriction, three special "system names" are set aside for such use. They are ULIB1, ULIB2 and ULIB3. In reality these are empty system libraries.

Use of any other "system names" for a user library local file name makes the system library with that name inaccessible until the local file is returned or renamed and the library set is redefined.

Libraries are searched in the order of their appearance on the latest LIBRARY statement.

A library set local to a load sequence may be specified with the 'LDSET,LIB.' statement.

A nonexistent or ill-formatted library produces a non-fatal error, but not until the loader tries to use the library.

Examples: Examples of the LIBRARY statement are found in the *Cyber Loader Reference Manual* (Publication No. 60344200), Section 7. Information on library creation can be found in the *User's Guide Supplement: LIBEDIT—Cyber Loader Libraries.*

7.10.2 MAP

The MAP statement defines the load map type in effect for the remainder of the job. A MAP statement may appear within a load sequence or by itself.

Load maps are written, by default, on OUTPUT for a batch job, MAPFILE for an interactive job. MAPFILE is not rewound between loads.

```
MAP,{ON|OFF|PART|FULL}.
```

Options selected may be one of the following:

OFF No map produced. This is the interactive default. (Corresponds to 'LDSET,MAP=N.')

PART Map includes block names and addresses and loader statistics. This is the batch default. (Corresponds to 'LDSET,MAP=SB.')

ON Map includes the above (PART) plus a cross-reference index of external references and entry points. (Corresponds to 'LDSET,MAP=SBX.')

FULL Map includes the above (ON), and in addition, the cross-reference index lists entry points that were never referenced. (Corresponds to 'LDSET,MAP=SBEX.')

Examples: Examples are available in the *Cyber Loader Reference Manual* (Publication No. 60344200), Section 2.

7.11 Field Length Control Statements

This section covers various control statements related to determining the amount of central memory needed for jobs.

First it is necessary to understand a number of terms.

Field length (FL) — The number of central memory (CM) words assigned to a job or needed by a program.

Job FL (JFL) — The field length at which a job is currently executing.

Maximum FL (MFL) — The largest value allowed for Job FL. If the job card CM parameter (or its default) is less than 100000_8, Maximum FL is initially set to 100000_8 or the Authorization File CM limit, whichever is smaller; if the job card CM parameter is greater than 100000_8, Maximum FL is initially set equal to the CM parameter. Maximum FL may be changed during a job by the MFL statement, but may not be raised above its initial value.

Starting Field Length (SFL) — This value is taken from the job card CM field length. The default CM entry from the authorization file is used if the job card CM field length is absent. Of course, SFL cannot exceed the Authorization File maximum CM entry.

User FL (UFL) — Field length used for "user loads." It initially equals the job card CM parameter (or its default). It may be changed during a job by the RFL statement, but may never exceed Maximum FL.

User load — Either:

a. load of a size-sensitive program from a library when AUTORFL is set to PART;

b. load of any program from a local file when REDUCE is set to OFF; or

c. load of any system program when AUTORFL is set to OFF.

Execution FL (EFL) — The address of the last CM word of a loaded program, rounded up to the next 100_8-word multiple by adding 102_8, then dropping the two low-order octal digits. For core-image modules, EFL refers to the root segment or main overlay only.

Highest High Address (HHA) — Applies to overlays only. HHA is the field length needed to load the largest possible combination of main, primary, and secondary overlays. It is present in the header of (0,0) overlays generated since July 31, 1976 (including those generated by 'NOGO,lfn.' at the end of a basic relocatable load).

Default FL (DFL) — For library programs only, the FL that is stored in the library directory. For complete listings of default FLs for all system and HAL routines execute 'HAL,INDEX.' (Note: This statement produces a large amount of output.) For selected FL requirements execute 'HELP,E*progname.'

Quick-running — System programs which execute at the current Job FL, if that is sufficient.

Size-sensitive — Library programs for which the amount of memory needed depends on the task they are doing, but cannot obtain the needed FL themselves. 'HAL,INDEX.' and 'HELP,E*progname.' indicate which system- and HAL-resident routines are size-sensitive. Programs on user libraries are declared size-sensitive by setting FLO=1 via LIBEDIT (see the *User's Guide Supplement: LIBEDIT—Cyber Loader Libraries*).

The following algorithm is used by the system to determine the Job Field Length.

1. If all programs were loaded from libraries, go to step 2; if at least one program was loaded from a local file, go to step 5.

2. If the program is quick-running and the current Job FL is big enough, Job FL is not changed. Otherwise, continue with step 3.

3. If AUTORFL is ON (the default condition), or if AUTORFL is PART and the program is not size-sensitive, go to step 4. Otherwise, set Job FL equal to User FL and stop.

4. Provisionally set Job FL equal to the larger of: (a) the program's Execution FL, or (b) the program's default FL. Go to step 8.

5. If REDUCE is ON (the default condition), go to step 6. If REDUCE is OFF, then set Job FL equal to User FL and stop.

6. If this is an absolute load (main overlay or root segment), go to step 7. If this is a relocatable load, then provisionally set the Job FL equal to the loaded program's Execution FL and go to step 8.

7. For a root segment, provisionally set Job FL equal to the root segment's Execution FL and go to step 8. For other absolute programs, if Highest High Address (HHA) is present in the header of the program, provisionally set the Job FL equal to HHA. Otherwise, provisionally set the Job FL to the larger of (a) the Execution FL of the program being loaded or (b) the User FL. In either case, continue with step 8.

8. If the provisional Job FL exceeds Maximum FL, set Job FL equal to Maximum FL.

Caution: If 'AUTORFL,OFF.' is used, the Job FL is set to the User FL and the above algorithm is not used. 'AUTORFL,OFF.' should only be used in cases where it is the only alternative for obtaining needed field length.

7.11.1 AUTORFL

The AUTORFL statement determines the field length (FL) at which size-sensitive library programs are executed. 'HAL,INDEX.' indicates which programs in system libraries and in the HUSTLER Auxiliary Library are size-sensitive.

```
AUTORFL,{ON|OFF|PART}.
```

ON Size-sensitive library programs are executed at their default FL. This is the default.

PART Size-sensitive library programs are executed at the user FL as defined by the last RFL statement, or by the job card CM parameter or its default.

OFF All programs executed at the user field length.

This option should be used with extreme caution, since it overrides all system field length settings. Its use should be limited to cases where it is the only alternative for obtaining needed field length.

Programs in user libraries may be made size-sensitive by using LIBEDIT to set FLO to 1 (see *User's Guide Supplement: LIBEDIT—Cyber Loader Libraries*).

7.11.2 MFL

The MFL statement is used to reset the job Maximum Field Length. The initial Maximum Field Length value is established at the beginning of the job as follows:

job card CM field or default	initial Maximum Field Length
$\leqslant 100000_8$	minimum (100000_8, A.F. max CM)
$>100000_8$	job card CM field

where 'A.F. max CM' is the authorization file maximum CM entry. The MFL control statement can change the maximum field length from its initial value:

```
MFL,nnnnnn.
```

nnnnnn an octal number specifying the new Maximum Field Length. This value must not exceed the initial MFL value. If nnnnnn is zero or omitted, the MFL is set to the initial MFL.

If the new MFL is less than the User's Field Length (UFL), the UFL is set to the MFL. The new maximum field length value may never exceed the initial MFL value. See the Job Memory Management examples in Section 7.11.5.

7.11.3 REDUCE

The REDUCE statement determines the execution field length (FL) of programs loaded from local files.

REDUCE[,ON|,OFF].

ON Programs loaded from local files execute at the smallest FL possible (rounded to the next 100_8 word multiple). This is the default, except in overlay loads, where REDUCE is ignored.

OFF Programs loaded from local files execute at the User FL, as defined by the last RFL statement or by the job card CM parameter or its default.

'REDUCE.' may occur within a load sequence, in which case it has effect only on that load.

The REDUCE statement is ignored in overlay loading; it is automatically set to OFF.

Certain kinds of programs should be run with REDUCE set to OFF. These are programs written in ALGOL, LISP, SIMULA, or SNOBOL.

7.11.4 RFL

The RFL statement resets the User Field Length (but does not change the current Job Field Length). The initial User Field Length is defined by the job card CM parameter or its default. The default at the beginning of an interactive session is 40000_8.

RFL,nnnnnn.

nnnnn an octal value specifying the new User Field Length. This must not exceed the job's Maximum Field Length.

See the Job Memory Management Examples in Section 7.11.5.

7.11.5 Job Memory Management Examples

In the following examples, assume that the Authorization File default CM is 45,000 (octal) and that the Authorization File maximum CM is 120,000 (octal).

1.	id.	job card	MFL is 100000 SFL is 45000 UFL is 45000
	FTN.	FTN compiles with the JFL set to 46000. The UFL remains at 45000.	
	LGO.	Loading is done at the UFL value of 45000.	

2. id,CM50000. job card MFL is 100000
SFL is 50000
UFL is 50000

FTN,OPT=2. FTN sets the JFL to 60000 and compiles the program, and then returns the JFL to 50000.

RFL,67000. UFL is 67000. This RFL is allowed as the MFL is 100000.

LGO. Loading is done at 67000.

3. id,CM60000. job card MFL is 100000
SFL is 60000
UFL is 60000

AUTORFL,PART.

MFL,70000. Reduce MFL to 70000.

COMPASS. COMPASS begins assembly at the UFL value of 60000. During the assembly COMPASS may request additional field length up to a maximum of 70000.

4. id,CM110000. job card MFL is 110000.
SFL is 110000.
UFL is 110000.

AUTORFL,PART.

HAL,SPSS. SPSS will be executed at a field length of 110000.

In the next example, assume that the Authorization File default CM is 54,000 (octal) and that the Authorization File maximum CM is 60,000 (octal).

5. id,CM50000. job card MFL is 60000
SFL is 50000
UFL is 50000

AUTORFL,PART.

COMPASS. COMPASS begins assembly at 50000. During the assembly it may request additional field length up to a maximum of 60000.

In the last example, assume that the Authorization File default CM is 43,000 (octal) and that the Authorization File maximum CM is 60,000 (octal).

6. id. job card MFL is 60000
SFL is 43000
UFL is 43000

FTN. FTN compiles with the JFL set to 46000. The UFL remains at 43000.

RFL,67000. This RFL will cause the job to abort since the MFL is only 60000.

Remember, if the Starting Field Length (SFL) is greater than 100,000 (octal), the SFL is the initial MFL. However, the MFL may be altered with MFL control statement as explained earlier.

7.12 Processing Options

The following are control statements whose functions do not fit more appropriately in any other category.

7.12.1 BANNER

BANNER is a control-statement-callable output utility that prints titles in large block letters. BANNER produces one page of output consisting of up to four ten-character lines.

```
HAL.BANNER[,O=outlfn][,string][,...].
```

O=outlfn the output file into which BANNER writes; the default is OUTPUT.

string character strings of length 0 to 10; may include alphanumeric characters, imbedded blanks, and special characters except comma, period, parenthesis, or dollar sign. Up to 4 strings may be specified. Any or all of the strings may be missing; a null string is taken as a blank line. A comma or parenthesis is taken as a delimiter; a period ends the statement.

The following special strings may be specified:

*D generates a banner line with the current date (mm/dd/yy).

*T generates a banner line with the current time (hh.mm.ss).

The job sequence number, the time and data are listed on a single line. BANNER does not connect OUTPUT or the file specified by the O=outlfn parameter.

7.12.2 COMMENT

The COMMENT statement has nothing to do with program execution or any other system task. It is useful only for inserting comments in the job dayfile. The comments will appear in the dayfile.

```
COMMENT, user comments.
```

user comments any string of characters (optional).

Example:

```
PNC
job card
PW=password
FTN.
COMMENT, PROGRAM IS NOW COMPILED.
COMMENT, READY TO LOAD AND EXECUTE.
LGO.
7/8/9
FORTRAN PROGRAM
6/7/8/9
```

7.12.3 COPIES

The COPIES statement is used by batch jobs to request additional copies of either the OUTPUT file or a punch file (i.e., PUNCH, PUNCHB, PUNCHC, PUNCH9, P80C), or any file given a print or punch disposition and later returned.

```
COPIES,{OUTPUT|PUNCH}[,n].
```

OUTPUT Print additional copies of the job's OUTPUT file.

PUNCH Punch additional copies of all punch files. This will affect the files PUNCH, PUNCHC, PUNCHB, P80C, and PUNCH9.

n Number of additional copies desired (optional) (1≤n≤63). The default is 1 additional copy. Note that 1 additional copy implies a total of 2 copies.

COPIES applies only to jobs printed or punched at the central site. Remote batch terminal users may direct job output to the central site with a 'DISPOSE,**,B.' control statement (see Section 7.4.1).

To request multiple copies of files other than those listed above, use the 'C=' parameter on the 'DISPOSE,lfn.' statement.

The total number of pages printed or cards punched is limited by the job card L and C parameters, respectively.

Example:

```
PNC
job card
PW=password
ATTACH,X,SPSSDATA.
HAL,SPSS,D=X.
COPIES,OUTPUT,2.
7/8/9
SPSS control cards
6/7/8/9
```

This example shows an SPSS job being run, and requests that a total of three copies of the output be produced.

7.12.4 DAYMSG

The DAYMSG statement determines what type of messages will be displayed in the dayfile (batch) or at the terminal (interactive). Two types of messages are displayed: dayfile messages and information messages (including "COMPILING...", "ASSEMBLING...", and "UPDATING...").

```
DAYMSG,{ON|PART|OFF}.
```

ON Interactive: Both dayfile messages and informative messages are displayed. This is the default at the start of a job.

Batch: dayfile messages are printed.

PART Interactive: Informative messages are suppressed; the only messages displayed are dayfile messages.

Batch: dayfile messages are printed (same as ON).

OFF Interactive: All dayfile messages for the terminal are suppressed.

Batch: All dayfile messages from central processor programs are suppressed; this includes all user programs and a vast majority of control-statement-callable system programs. Accounting, RP, and NL messages are not suppressed.

Caution: Setting DAYMSG to OFF can suppress important error messages. It should be used with care.

7.12.5 LIMIT

The LIMIT statement is used to reset the disk storage limit during job execution.

```
LIMIT,nnn.
```

nnn the new disk storage limit. This is an octal number representing the number of units of disk storage (one unit equals 100_8 words of central memory).

The disk storage associated wit ı a job is the total disk space occupied by all files currently assigned to the job, excluding permanent files attached with read-only permission. The maximum disk storage limit permitted for the PN may be displayed using AUTHORF.

Example:

Normally, a job is assigned as much disk storage as it needs; however a user may want to limit the maximum disk storage that should be assigned, for example, during a "dry run" phase when large amounts of output would indicate program errors. In the example, the user limits disk storage to 30000_8 words; the job will terminate when that limit is reached.

```
PNC
job card
PW=password
LIMIT,300.
HAL,SPSS.
7/8/9
SPSS job
6/7/8/9
```

7.12.6 PAUSE

The PAUSE control statement will cause a job to display a message of up to 20 characters to the operator. The job will resume execution when the operator tells it to "GO."

```
HAL,PAUSE[,message].
```

Only the first 20 characters of the message will be sent. The default message is "PAUSE—AT USER REQUEST."

All messages will be prefixed by the word "PAUSE."

Example:

The PAUSE statement is often used to work around incompatible labels on magnetic tapes, such as when the visual reel name (VRN) written on the internal tape label does not match the VRN pasted on the tape (see Section 6.15.2). In the following job, the user notifies the operation of the incompatibility:

```
PNC
job card
PW=password
HAL,PAUSE,USE LY100 FOR LY102.
REQUEST,TAPE1,VRN=LY102.
FTN.
LGO.
6/7/8/9
```

7.12.7 RERUN

The RERUN statement sets a flag which indicates whether the job may be restarted by the operator. RERUN is meaningful only for batch jobs.

RERUN statements take effect as they are encountered.

```
RERUN[,ON|,OFF].
```

ON The operator may restart the job. This is the default.

OFF The operator may not restart the job.

The RERUN flag is set "ON" at the start of a job. Certain system routines set the RERUN flag to OFF without notification to the user. This occurs, for example, when a successful plot is generated.

The usual reasons for restarting a job are production "shut-down" or system difficulties. Restarting a job involves the following steps:

1. The INPUT file is returned to the input queue.

2. The partial dayfile is copied to a new "pre-output" file and the file OUTPUT is destroyed. When the job starts executing again, the "pre-output" file becomes the OUTPUT file. Thus

the final job output contains two dayfiles, one for the partial run and one for the complete run.

3. Permanent files are handled as if the job had terminated normally.

4. All other files are dropped, regardless of name or disposition code.

No other reinitialization is performed. For example, if a job has written on a magnetic tape, it is not erased; permanent files cataloged before the job is rerun are not purged.

'RERUN,OFF.' should be used under the following conditions:

1. A change is made to the file INPUT (a practice not recommended in any event).

2. A permanent file is created, modified or extended.

3. A magnetic tape is both read and rewritten during the same job.

The 'RERUN,OFF.' statement should appear before any of the above actions are taken.

The rerun flag may also be set by a COMPASS macro (see Section 8.5.14) or an FTN- and COBOL-callable subroutine (see *User's Guide Supplement: FORTRAN Extended Library Routines*).

7.12.8 SWITCH

This statement is used to change the setting of a sense switch.

```
SWITCH,n[,ON|,OFF].
```

n Sense switch number; 1≤n≤6.

ON Switch n is turned on.

OFF Switch n is turned off.

If neither ON nor OFF is specified, switch n is toggled; that is, it is turned on if it was off, and vice versa.

By default, all sense switches are off at the beginning of a job.

7.13 Debugging

This section describes control statements used to obtain information helpful in determining the cause of program errors. The main sources of debugging information are:

1. Dayfile—a listing of the control statements executed, system diagnostics, and accounting messages.

2. Dump—a printout of the current contents of the central processor registers and selected memory areas.

3. Load map—a listing showing the location of each subprogram and common block within the central memory field length assigned to the job.

4. Reference map—a compiler-generated map of program elements. Together, the reference map and load map enable the programmer to find the address of a particular variable or program statement.

This section deals primarily with dumps. For a description of the load map, see the CDC *Cyber Loader Reference Manual* (Publication No. 60344200). For a description of the reference map, see the appropriate programming language reference manual. In addition to the reference map, the programming language may provide options for tracing program flow, checking array references, and detecting illegal data values. These facilities are also described in each programming language reference manual.

Some other debugging aids, not described in this section include:

EXIT This card is needed to continue job processing after an abnormal termination. See Section 7.1.3.

RECOVR This FORTRAN library subroutine allows programs to continue execution after the occurrence of fatal errors. See *User's Guide Supplement: FORTRAN Extended Library Routines*.

REPRIEVE This macro performs the function of RECOVR for COMPASS programs. See Section 8.5.17.

7.13.1 DAYFILE

The DAYFILE statement is used to save dayfile messages on a local file and to display selected line ranges from this file at an interactive terminal or print them in the OUTPUT file. Although DAYFILE is intended primarily to allow interactive users to examine their dayfiles, it can also be useful in batch jobs, especially batch jobs submitted from remote sites or from interactive terminals.

The DAYFILE statement has two formats.

Format 1: DAYFILE[,F].

Format 2: DAYFILE,fromline,toline[,F].

F full option; all specified lines are displayed. If omitted, the CP, PP, NL, RP, and other accounting messages issued in the last 11 lines are not displayed.

fromline
toline specify the line range to be displayed or printed. The first line of the current copy of DAYF is line 1. If the last number is not known, any number larger than the number of dayfile lines expected to be issued may be specified.

Format 1 saves on file DAYF all dayfile messages accumulated since the previous format 1 DAYFILE command or start of job. It then displays or prints up to 11 lines of the most recent messages.

Format 2 is used to display messages issued prior to the last 11 lines. Before any DAYFILE command in format 2 can be used, at least one DAYFILE command in format 1 must be given. Format 2 reads only the current copy of DAYF; it does not update this file with new messages.

Cautions:

1. Note that whenever 'DAYFILE.' or 'DAYFILE,F.' is executed, the messages saved by the previous Format 1 DAYFILE command are lost. A new copy of DAYF is written and the new dayfile lines are assigned line numbers starting from 1.

2. The local file DAYF cannot simply be copied to an output file. For obscure reasons, each PRU (64-word block) of DAYF contains 48 words of the user's dayfile and 16 words of somebody else's dayfile. The DAYFILE routine sorts out this mixture and prints only the lines belonging to the user.

3. When a batch job executes format 1 of the DAYFILE statement, the messages copied to DAYF are not printed in the dayfile at the end of the job. If the user wants a copy of the entire dayfile printed in the job output, the format 1 DAYFILE command must be followed by a format 2 command, such as 'DAYFILE,1,999,F.' To get the most complete dayfile, these control statements should be at the end of the control section.

4. Because the accounting summary is not generated until job termination, these messages cannot be copied to DAYF.

5. Used interactively, DAYFILE connects file OUTPUT.

Example 1:

Below are samples of the DAYFILE command used interactively. (User type-ins are shaded.) Batch output would be identical.

```
OK-dayfile.
      33 .14.26.24.ZZZZPRG - CYCLE 01, MTLY017PART2
      34 .14.26.24.FILE ATTACHED
      35 .14.26.24.ZZZZPRG - CYCLE 01, MTLY01, MTLY017PART2
      36 .14.26.24.FILE SUCCESSFULLY PURGED
      38 .14.27.37.HAL,STATUS.
      39 .14.27.38.ZZZZMPL - CYCLE 01, HAL 5, MPL
      40 .14.27.38.FILE ATTACHED
         STOP
```

```
OK-dayfile,25,33,f.
     25 .14.24.44.NL    34200
     26 .14.24.44.RP  000000001637  000000043107
     27 .14.24.45.NL    34200
     28 .14.24.45.RP  000000001641  000000043260
     29 .14.24.45.NL    52200
     30 .14.24.45.RP  000000001643  000000043260
     31 .14.26.23.CP-PP SEC.    11.969-    68.416   $    3.32
     32 .14.26.23.PNPURGE,PFN=MTLY017PART2
     33 .14.26.24.ZZZZPRG - CYCLE 01, MTLY017PART 2
        STOP
OK-dayfile,f.
      1 .14.29.27.CP-PP SEC.    12.263-    74.166   $    3.49
      2 .14.29.27.DAYFILE,25,33,F.
      3 .14.30.03.CP-PP SEC.    12.292-    74.594   $    3.52
        STOP
OK-
```

Example 2:

When a batch job is submitted from an interactive session or from a remote batch site, the user may wish to catalog the job dayfile (and possibly the job output, too), so that the dayfile can be examined at an interactive terminal to determine whether the job executed properly before printing or picking up the output. Here is an example:

```
PNC
job card
PW=password
.
.
EXIT,S,C.
DAYFILE.
CATALOG,DAYF,JALDAYF,RP=2.
DAYFILE,1,999,F.
7/8/9
.
.
.
6/7/8/9
```

The 'EXIT,S,C.' control statement forces processing of the DAYFILE statement immediately following. The first DAYFILE statement creates DAYF; the second one prints the contents in the job output. Later, at an interactive terminal, the user could attach JALDAYF as local file DAYF and execute format 2 of DAYFILE to examine it.

```
RETURN,DAYF.
ATTACH,DAYF,JALDAYF.
DAYFILE,1,999,F.
.
.
.
PURGE,DAYF.
RETURN,DAYF.
```

7.13.2 DMP

The DMP statement generates a printed copy of all or part of the job field length. Each word of memory is displayed as 20 octal digits. The dump is printed four words per line with the address of the first word at the beginning of the line. Printing of a central memory word is suppressed when that word is identical to the last word printed. When the next non-identical word is encountered, its address is printed and marked by an equal sign (=).

DMP always writes to the end of file OUTPUT. If the job is interactive, OUTPUT is first disconnected.

The DMP statement has three forms, which can be represented as:

```
DMP.
DMP,fwadump,lwadump.
DMP,lwadump.
```

fwadump the starting address, relative to the job field length. The first word of the field length is address 0.

lwadump the last word address of the dump, relative to the job field length. The form 'DMP,lwadump.' is equivalent to 'DMP,0,lwadump.' The dump is always restricted to the user's field length. DMP automatically corrects the dump limits if the specified limits are out of bounds.

The form 'DMP.' produces a special format known as the standard, or exchange package, dump. The standard dump displays the contents of the central processor registers, the first 100 (octal) words of the field length, and 100 (octal) words before and after the address of the last instruction executed. If the DMPX option is ON (see Section 7.13.3), a standard dump will be produced automatically when the job terminates abnormally; this is the default condition for central site batch jobs. 'DMPX,OFF.' is the default for remote batch jobs. A sample standard dump appears in Figure 7.5.

Item 1, labeled P, is the program address register. It contains the address of the next instruction word that would have been executed if the program had not terminated. This register may be zero if a mode error occurs, in which case the P address is given in bits 47-30 of location 0 (see first line of item 13).

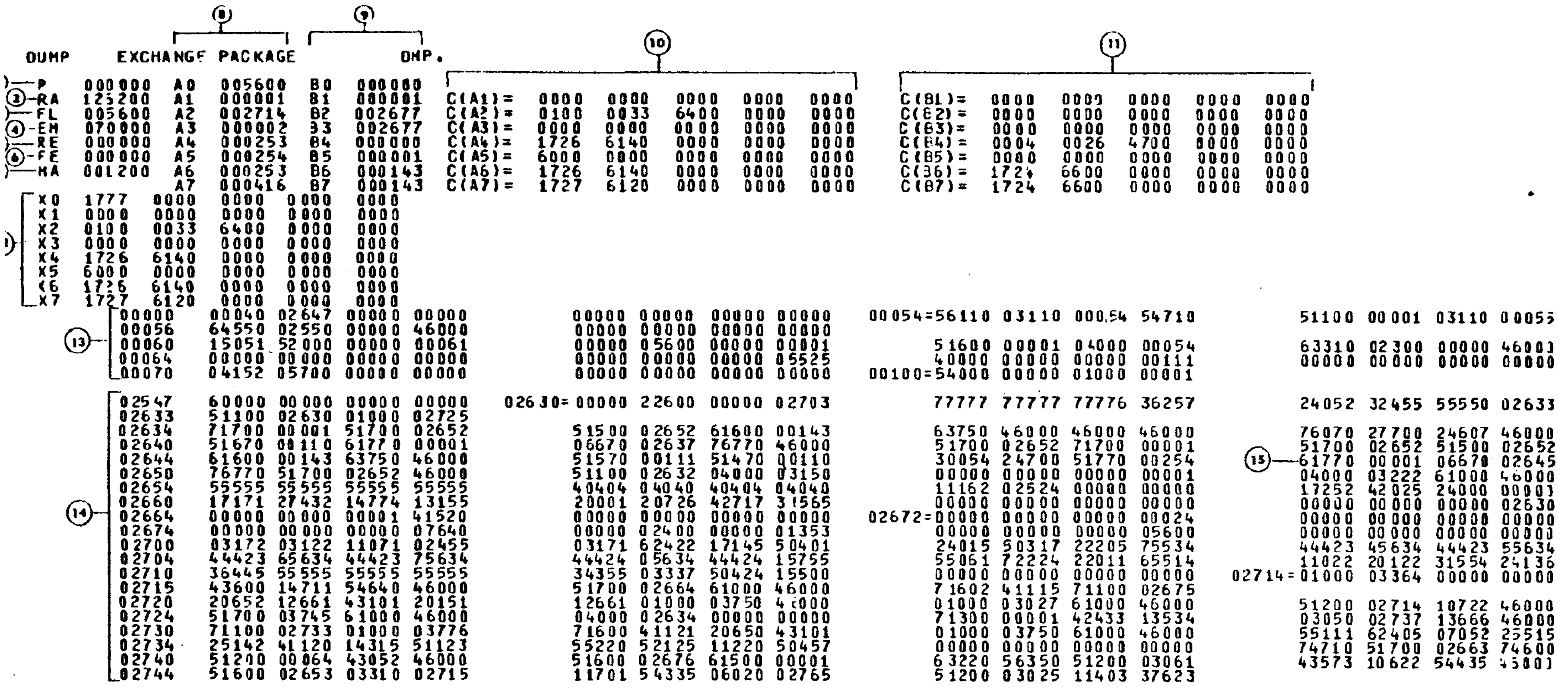
Item 2, labeled RA, is the reference address. This is the absolute address of the first word of the user's central memory area.

Item 3, labeled FL, is the field length of the job. This is the number of central memory words currently assigned to the job.

Item 4, labeled EM, is the exit mode register. It specifies error conditions on which the central processor will terminate execution, as described for the MODE statement in Section 7.13.5. The default condition, shown here, is 7, which causes termination should any of the three types of CPU-detected errors occur.

Items 5 and 6, labeled RE and FE, are the reference address and field length of the user's extended core storage (ECS) area. Users cannot access ECS at MSU, so these values are always zero.

Item 7, labeled MA, is the monitor address. This is used by the system monitor when swapping the central processor between two jobs. It is of no importance to the user.



2
4
6
8
9
10
11
13
14
15

```
DUMP      EXCHANGE PACKAGE                DMP.

  P   000000  A0  005600  B0  000000
  RA  125200  A1  000001  B1  000001  C(A1)=  0000  0000  0000  0000  0000   C(B1)=  0000  0000  0000  0000  0000
  FL  005600  A2  002714  B2  002677  C(A2)=  0100  0033  6400  0000  0000   C(B2)=  0000  0000  0000  0000  0000
  EM  070000  A3  000002  33  002677  C(A3)=  0000  0000  0000  0000  0000   C(B3)=  0000  0000  0000  0000  0000
  RE  000000  A4  000253  B4  000000  C(A4)=  1726  6140  0000  0000  0000   C(B4)=  0004  0026  4700  0000  0000
  FE  000000  A5  000254  B5  000001  C(A5)=  6000  0000  0000  0000  0000   C(B5)=  0000  0000  0000  0000  0000
  MA  001200  A6  000253  B6  000143  C(A6)=  1726  6140  0000  0000  0000   C(36)=  1724  6600  0000  0000  0000
              A7  000416  B7  000143  C(A7)=  1727  6120  0000  0000  0000   C(B7)=  1724  6600  0000  0000  0000
  X0  1777  0000  0000  0000  0000
  X1  0000  0000  0000  0000  0000
  X2  0100  0033  6400  0000  0000
  X3  0000  0000  0000  0000  0000
  X4  1726  6140  0000  0000  0000
  X5  6000  0000  0000  0000  0000
  <6  1726  6140  0000  0000  0000
  X7  1727  6120  0000  0000  0000
      00000  00040 02647 00000 00000         00000 00000 00000 00000   00054=56110 03110 00054 54710         51100 00001 03110 00055
      00056  64550 02550 00000 46000         00000 00000 00000 00000
      00060  15051 52000 00000 00061         00000 05600 00000 00001         51600 00001 04000 00054         63310 02300 00000 46001
      00064  00000 00000 00000 00000         00000 00000 00000 05525         40000 00000 00000 00111         00000 00000 00000 00000
      00070  04152 05700 00000 00000         00000 00000 00000 00000   00100=54000 00000 01000 00001

      02547  60000 00000 00000 00000  02630= 00000 22600 00000 02703         77777 77777 77776 36257         24052 32455 55550 02633
      02633  51100 02630 01000 02725
      02634  71700 00001 51700 02652         51500 02652 61600 00143         63750 46000 46000 46000         76070 27700 24607 46000
      02640  51670 00110 61770 00001         06670 02637 76770 46000         51700 02652 71700 00001         51700 02652 51500 02652
      02644  61600 00143 63750 46000         51570 00111 51470 00110         30054 24700 51770 00254         61770 00001 06670 02645
      02650  76770 51700 02652 46000         51100 02632 04000 03150         00000 00000 00000 00001         04000 03222 61000 46000
      02654  55555 55555 55555 55555         40404 04040 40404 04040         11162 02524 00000 00000         17252 42025 24000 00001
      02660  17171 27432 14774 13155         20001 20726 42717 31565         00000 00000 00000 00000         00000 00000 00000 02630
      02664  00000 00000 00001 41520         00000 00000 00000 00000   02672=00000 00000 00000 00024         00000 00000 00000 00000
      02674  00000 00000 00000 07640         00000 02400 00000 01353         00000 00000 00000 05600         00000 00000 00000 00001
      02700  03172 03122 11071 02455         03171 62422 17145 50401         24015 50317 22205 75534         44423 45634 44423 55634
      02704  44423 65634 44423 75634         44424 05634 44424 15755         55061 72224 22011 65514         11022 20122 31554 24136
      02710  36445 55555 55555 55555         34355 03337 50424 15500         00000 00000 00000 00000   02714=01000 03364 00000 00000
      02715  43600 14711 54640 46000         51700 02664 61000 46000         71602 41115 71100 02675
      02720  20652 12661 43101 20151         12661 01000 03750 46000         01000 03027 61000 46000         51200 02714 10722 46000
      02724  51700 03745 61000 46000         04000 02634 00000 00000         71300 00001 42433 13534         03050 02737 13666 46000
      02730  71100 02733 01000 03776         71600 41121 20650 43101         01000 03750 61000 46000         55111 62405 07052 25515
      02734  25142 41120 14315 51123         55220 52125 11220 50457         00000 00000 00000 00000         74710 51700 02663 74600
      02740  51200 00064 43052 46000         51600 02676 61500 00001         63220 56350 51200 03061         43573 10622 54435 45801
      02744  51600 02653 03310 02715         11701 54335 06020 02765         51200 03025 11403 37623
```

Figure 7.5: Standard Dump

Item 8 is a column showing the contents of the A registers. A1 through A5 are linked to X1 through X5 such that when an address is put into the A register the contents of that memory location are loaded into the corresponding X register. A6 and A7 are linked to X6 and X7 in a converse fashion: when an address is put into the A register, the content of the corresponding X register is stored at that location. A0 and X0 are not linked and are used, instead, as "scratch pads."

Item 9 is a column showing the contents of the B registers. The B registers are typically used to hold loop indexes and to retain memory addresses.

Item 10 shows the memory locations referenced by the A registers, except A0. This is not produced if the location referenced by an A register is outside the job's field length.

Item 11 shows the memory locations referenced by the B registers, except B0. This item is not produced if the location referenced by a B register is outside the job's field length.

Item 12 shows the contents of the X registers. Registers X1 through X5 typically hold operands read from central memory for use in calculations, and registers X6 and X7 typically hold results of calculations to be sent to central memory.

Item 13 shows locations 0 through 100 of the job field length. This area is used for communication between the job and the operating system.

Item 14 shows the 100 (octal) locations preceding and following the P address. In the sample, the program terminated at location 2647, so locations 2547 through 2747 are displayed. (Although the P register is 0, the P address is given in bits 47-30 of location 0.) Note that contiguous locations with identical contents are not displayed. For example, locations 2547 through 2627 contain 60000 00000 00000 00000, and locations 2665 through 2671 and 2712 through 2713 contain zero.

Item 15 is the word pointed to by the P address, the instruction word that would have been executed next.

A dump may be obtained at either normal or abnormal program termination. To obtain a dump at abnormal termination, the DMP statement must follow an EXIT statement. If used in interactive mode, the EXIT and/or DMP commands **must follow on the same line** as the command that initiates program execution, as in the following:

```
FTN.
LGO. EXIT. DMP.
```

The commands may also be placed on separate lines in a control statement file and executed via EXEC (see Section 7.1.2). (Interactive users may find the DEBUG statement more useful; see the *Interactive System User's Guide*, Chapter 6.)

7.13.3 DMPX

The DMPX statement enables or inhibits the automatic generation of a standard dump in the event of an abnormal termination. The standard dump is a printed output of the central processor registers, location 0-100_8 of the job field length, and the 100_8 memory locations before and after the last instruction executed. A sample is described in Section 7.13.2 and shown in Figure 7.5.

The format of the DMPX statement is

```
DMPX,{ON|OFF}.
```

ON enables the automatic dump. This is the normal condition for batch jobs submitted at the Central Site and batch jobs submitted via DISPOSE from interactive terminals.

OFF suppresses the automatic dump. This is the normal condition for jobs submitted from remote batch terminals.

DMPX is ignored in an interactive session.

7.13.4 ERRS

The ERRS statement is used to scan a file for error diagnostic messages. Currently, ERRS can process listings from FTN (except FTN,TS), SPSS, COBOL, COMPASS, and UPDATE. For each listing that contains an error message, ERRS prints an error summary consisting of:

1. the program name.
2. the number of informative error messages.
3. a listing of the lines in error (optional).
4. a list of the fatal error messages and the sequence number(s) of the line(s) to which each message applies.

Although it is intended primarily for interactive use, ERRS works equally well in batch mode. Note, however, the difference in default options, as described below.

ERRS[,optional parameters].

ALL instructs ERRS to include non-fatal diagnostic messages in the summaries. The default is fatal error diagnostics only.

F instructs ERRS to print a full summary, which includes a listing of all lines in which errors occurred. This is the default for batch use. The interactive default is S.

I=inlfn specifies the input file, which ERRS will search for program listings. This file will be rewound before and after ERRS processing; in interactive use, it will also be disconnected. The default input file is always OUTPUT.

NA do not abort if errors occur.

NI eliminates the informative errors message if no fatal error diagnostics are found. The NI parameter is ignored if the ALL parameter is specified.

NS eliminates the search for sequence numbers within the listing and uses a line count relative to the start of the listing instead. This eliminates the second pass for most compiler listings and reduces execution time accordingly. A line count is also used if the NS parameter is omitted and the listing does not contain sequence numbers. Valid sequence numbers include EDITOR or UPDATE numbers (columns 73-90) and COBOL sequence numbers (columns 1-6).

O=outlfn specifies the output file, on which ERRS will write the error summaries. From batch, the default is OUTPUT. In interactive use, the default is TTYTTY, or if the keyword O appears alone, the connected file ZZZZOT. **Caution:** If the input and output files are the same, which is the default for batch users, the program listings, as well as any other information on the input file, will be destroyed.

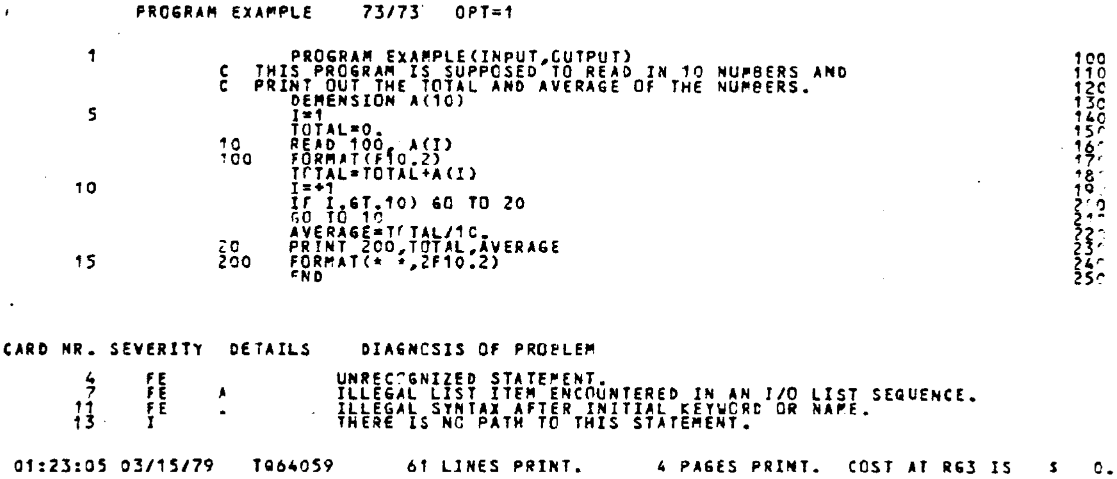
PG=n specifies the maximum number of pages that ERRS is to search for a recognizable listing. That is, if ERRS scans n pages without finding the start of a listing, it will terminate. The default page limit is 10. If page headers are absent, 80 lines are treated as a page.

S instructs ERRS to print a short summary, which gives only the diagnostics and the line numbers to which they apply (i.e., the source lines are not printed). This is the default for interactive usage. (Exception: When ERRS is run from an interactive terminal, the default output option for COMPASS error summaries is to list the lines in error and omit the error messages, rather than follow the normal S option. If F or S is explicitly declared, however, COMPASS listings will be treated the same as any other listing.) The batch default is F.

Caution: If ERRS finds any fatal error messages, it will end with an abort request after completing the error summaries (unless the NA parameter is specified). If the user wants to continue job execution in such cases, an 'EXIT.' statement must follow the ERRS statement.

Example:

The FTN listing of an example program which contains several errors is shown below. Following it are sample outputs from ERRS illustrating the effects of various control statement options.



```
            PROGRAM EXAMPLE     73/73   OPT=1

     1                   PROGRAM EXAMPLE(INPUT,CUTPUT)                                          100
                  C  THIS PROGRAM IS SUPPOSED TO READ IN 10 NUMBERS AND                         110
                  C  PRINT OUT THE TOTAL AND AVERAGE OF THE NUMBERS.                            120
                         DEMENSION A(10)                                                        130
     5                   I=1                                                                    140
                         TOTAL=0.                                                               150
                  10     READ 100, A(I)                                                         160
                  100    FORMAT(F10.2)                                                          170
                         TOTAL=TOTAL+A(I)                                                       180
    10                   I=+1                                                                   190
                         IF I.GT.10) GO TO 20                                                   200
                         GO TO 10                                                               210
                         AVERAGE=TOTAL/10.                                                      220
                  20     PRINT 200,TOTAL,AVERAGE                                                230
    15            200    FORMAT(* *,2F10.2)                                                     240
                         END                                                                    250


CARD NR. SEVERITY  DETAILS    DIAGNOSIS OF PROBLEM

     4     FE                    UNRECOGNIZED STATEMENT.
     7     FE       A            ILLEGAL LIST ITEM ENCOUNTERED IN AN I/O LIST SEQUENCE.
    11     FE       .            ILLEGAL SYNTAX AFTER INITIAL KEYWORD OR NAME.
    13     I                     THERE IS NO PATH TO THIS STATEMENT.

 01:23:05 03/15/79   TQ64059       61 LINES PRINT.      4 PAGES PRINT.  COST AT R63 IS   $   0.
```

"ERRS,S." produces:

```
1 INFO ERRORS IN EXAMPLE
FE UNRECOGNIZED STATEMENT.
  130
FE ILLEGAL LIST ITEM ENCOUNTERED IN AN I/O LIST SEQUENCE.
  160
FE ILLEGAL SYNTAX AFTER INITIAL KEYWORD OR NAME.
  200
```

"ERRS,F,ALL,NS." produces:

```
1 INFO ERRORS IN EXAMPLE
ERRORS WERE FOUND IN THE FOLLOWING LINES:
4= DIMENSION A(10)
7=10 READ 100, A(I)
11= IF I.GT.10) GO TO 20
13= AVERAGE=TOTAL/10.

FE UNRECOGNIZED STATEMENT.
  4
FE ILLEGAL LIST ITEM ENCOUNTERED IN AN I/O LIST SEQUENCE.
  7
FE ILLEGAL SYNTAX AFTER INITIAL KEYWORD OR NAME.
  11
I THERE IS NO PATH TO THIS STATEMENT.
  13
```

7.13.5 MODE

The MODE statement allows a job to continue processing after encountering specified mode errors. Errors which can be ignored are:

> Reference to an address outside the field length of the job; such an address may be generated during loading if a nonexistent program is referenced.
>
> Reference to an operand for floating point arithmetic which has an indefinite value.
>
> Reference to an operand for floating point arithmetic which has an infinite value.

Normally, these errors will terminate processing; by using the MODE statement, however, any or all can be ignored as halt conditions, so that processing continues until another type of error is encountered that terminates the job, or until all control statements are executed.

The MODE control statement is of the following form:

    MODE,n.

n   a number specifying the halt conditions:

    0   No halt occurs; all mode errors are ignored.

1	Only if address is out of range (mode 1)
2	Only if operand is infinite (mode 2)
3	If address is out of range or operand is infinite (mode 1 or 2)
4	Only if operand is floating point number of indefinite value (mode 4)
5	If address is out of range or operand is floating point number of indefinite value (mode 1 or 4)
6	If operand is infinite or a floating point number of indefinite value (mode 2 or 4)
7	If operand is infinite or a floating point number of indefinite value or address is out of range (mode 1, 2, or 4). This is the default.

Interactive and batch usage of the MODE statement is the same. Once a MODE statement is encountered, it will remain in effect until the job terminates or another MODE statement is encountered.

Any MODE value that permits processing to continue regardless of a reference to an out of range address should be used with great caution. Resulting output will probably have no value. Under such conditions, an attempt to write outside FL appears to complete normally; however, no writing is done. When an attempt is made to read outside FL, zero is returned to the X register specified.

Example:

The following statement will permit processing to continue if a referenced address is out of range of the job field in central memory. Processing will halt however, if an infinite operand or a floating point operand of indefinite value is referenced.

```
MODE,6.
```

7.13.6 TRAP

TRAP is a debugging aid for use with the Cyber loader. The TRAP statement causes the TRAP routine to be loaded and become applicable to the next relocatable load sequence. The TRAP statement should immediately precede the load sequence to which it applies. The TRAP routine reads and interprets directives from the input file and causes TRAPPER, an execution time routine, to be loaded with the user's program. TRAPPER is the first program loaded. The output is written on an output file; this file should not be used as an output file by the trapped program. List output consists of a listing of directives and the resulting dumps.

```
TRAP[,I=inlfn][,L=listlfn].
```

I=inlfn the local file from which the directives will be read. The default is INPUT.

L=listlfn the local file on which the directives and dumps will be written. The default is TRAPS.

The general job format is as follows:

```
PNC
job card
PW=password
FTN.
TRAP.
LGO.
EXIT,S.
DISPOSE,TRAPS,PR.          dispose TRAP output to printer.
7/8/9
FORTRAN program
7/8/9
TRAP directives
6/7/8/9
```

In a TRAP routine, the user can request a FRAME, or snapshot dump, of selected areas of memory whenever a specified instruction is executed. The code can also be "tracked," using TRACK, dumping each register and memory location whenever it is changed.

FRAME Output:

FRAME output consists of a dump of all the registers (if requested; if not, only the contents of the P register appear in the dump), and a core dump of the area specified in the directives. Both the octal and Display code representations of the area are included in the core dump.

TRACK Output:

Output from TRACK consists of a dump of any registers or memory locations changed by the instruction, the COMPASS image of the instruction, and full register dumps at the beginning and the end of each range.

For a complete description of the FRAME and TRACK directives, see the CDC *Cyber Loader Reference Manual* (Publication No. 60344200), Chapter 6.

7.14 Permanent File Utilities

This section briefly describes permanent file utilities. A more complete discussion of permanent files is found in Chapter 5.

7.14.1 ATTACH

The ATTACH control statement makes an existing permanent file available to a job by assigning a local file name (lfn) to it and specifying whatever passwords may be required for the type(s) of access desired. After a permanent file has been attached, it is referred to by the local file name. The statement is given as follows:

```
ATTACH,lfn,pfn[,CY=cy][,MR=n][,PW=pw[,...]].
```

lfn the local file name to be given to the permanent file. This must be the first parameter.

pfn the permanent file name. This must be the second parameter.

CY=cy the optional cycle number. If omitted, the highest cycle number that exists is attached.

MR=n If n is 0, this parameter is ignored. If n is not 0, multiple read access is allowed (see Section 5.2.2).

PW=pw an optional list of up to 5 passwords for the desired types of access (see Section 5.1.4). Note that it is not necessary to indicate which password allows which type of access. If no passwords are specified, permission is granted for those types of access for which passwords were not defined when the file was cataloged. A turnkey password, if specified on the CATALOG statement, must appear on the ATTACH statement.

Note: The ATTACH control statement allows processing of continuation cards in batch mode. The parameter list may be separated after any comma and continued on the next card.

If an ATTACH request cannot be fulfilled because the file is assigned to another job, the following will occur: if the job is batch, it is swapped out and will wait for the file to become available; if the job is interactive, the ATTACH request is aborted and a message is printed at the user's terminal.

Cautions:

1. The following conditions cause fatal errors:

   a. The local file name is already in use by the job.

   b. The permanent file name or specified cycle does not exist.

   c. The permanent file is already attached to the job.

   d. If more than one interactive job requires concurrent read access on a file not set up for multi-read access.

e. If a file is cataloged for multi-read access by specifying Modify, Extend and Control passwords and the interactive job desiring concurrent access specifies any of these passwords.

2. In interactive mode, the ATTACH control statement must be terminated with a period.

Examples: See Section 5.2.2 for further discussion and examples.

7.14.2 CATALOG

The CATALOG control statement changes a temporary disk file to a permanent file by assigning it a permanent file name and optional passwords, cycle number, and retention period.

After a file has been "cataloged," subsequent jobs are able to access the file by means of the ATTACH statement. The file will remain cataloged until purged by the user or purged by the Computer Laboratory when:

1. the retention period has expired,
2. the user's PN has expired, or
3. the user's dollar balance has become zero or negative.

The user is charged for permanent file storage based on the size of the file and the period of storage.

The CATALOG statement has the form:

```
CATALOG,lfn,pfn[,optional parameters].
```

lfn the local file name of the file being cataloged. This must be a temporary disk file. This must be the first parameter.

pfn the permanent file name to be assigned to the file (1-40 alphabetic or numeric characters). This name is used when subsequent jobs access the file via the ATTACH statement; the ATTACH statement may assign any local file name to the permanent file. This must be the second parameter.

BC=n If n is 1, the Computer Laboratory will maintain a copy of the file on a backup tape. This is the default. If n is 0, the Computer Laboratory will not maintain a backup copy of the file.

CN=cn Control password (1-9 alphanumeric characters). Controls ability to purge the file and to create additional cycles of the file.

CY=cy a cycle number from 1 to 63. Up to five cycles of a particular permanent file name may be cataloged. The default is 1.

EX=ex Extend password (1-9 alphanumeric characters). Controls ability to add data to the end of the file.

ID=id a 1-9 character identifier to help the user identify files listed within a PFLIST report.

MD=md Modify password (1-9 alphanumeric characters). Controls ability to rewrite existing parts of the file.

MR=n If n is not 0, the file will have Read only permission after the CATALOG is completed, and will be accessible to other users (multi-read access). If n is 0, the permanent file will remain attached with all permissions and must be returned and attached with the proper permissions to obtain multi-read access. MR=0 is the default.

PW=cn,tk specifies passwords necessary to catalog additional cycles of a permanent file name. If Control and/or Turnkey passwords were specified on previous cycles, they must appear here. Up to two passwords may be necessary.

RD=rd Read password (1-9 alphanumeric characters). Controls ability to read the file.

RP=rp the retention period in days ($0 \leq rp \leq 999$). If rp is 0, the file is purged at the end of the day. If rp is 999, the file is retained indefinitely. The default is 15 days.

TK=tk Turnkey password (1-9 alphanumeric characters). Controls ability to attach the file.

If a particular type of password is not specified, the associated type of access is always permitted when the file is later attached.

Note: The CATALOG control statement allows processing of continuation cards in batch mode. The parameter list may be separated after any comma and continued on the next card.

Cautions:

1. The file being cataloged must be a temporary disk file.

2. Only five, uniquely numbered, cycles of a permanent file may be cataloged at any one time.

3. The TK, RD, EX, MD, and CN passwords cannot be added or altered when cataloging additional cycles of a permanent file.

4. If the user attempts to catalog a duplicate permanent file name, a duplicate cycle number, or too many cycles of a particular file, the system will generate a unique name by adding a digit prefix to the permanent file name, then catalog the file using the new name and the indicated cycle number.

5. To purge and recatalog a permanent file (possibly to change the name, retention period, or passwords), all associated passwords must be specified when the file is attached.

6. Cataloging does not alter the contents or position of the file; it does not cause the file to be copied.

7. Multiple-read access (read access by more than one job at the same time) can be obtained only by specifying the EX, MD, and CN passwords when the file is cataloged, or by specifying the MR parameter on the CATALOG or ATTACH statements.

8. Information subsequently copied to the end of a permanent file is lost unless an EXTEND is issued (see Section 7.14.3).

9. In interactive mode, the CATALOG control statement must be terminated with a period.

Examples: See Section 5.2.1 for further discussion and examples.

7.14.3 EXTEND

The EXTEND control statement allows the size of a permanent file to be increased by making permanent any information that has been written to the end of an attached permanent file. Without the EXTEND statement, any appended information would be dropped when the file was returned or the job ended. The permanent file must be attached with Extend permission in order to use the EXTEND statement.

```
EXTEND,lfn.
```

lfn the local file name of the file to be extended.

Caution: Extend permission allows the user to overwrite the last PRU of a permanent file. The effects of overwriting are permanent even if the EXTEND statement is not used.

Examples: See Section 5.2.4 for further discussion and examples.

7.14.4 PFDUMP

The PFDUMP utility allows users to dump specified permanent files from disk or PF dump tapes onto a new dump tape. This allows the user to merge files from disk and existing dump tapes on a new dump tape. It also provides an output file describing the action taken by PFDUMP on each permanent file.

The PFDUMP control statement is of the form:

```
PFDUMP[,optional parameters].
```

The optional parameters are divided into five functional groups:

Tape specification
Disk specification
Output file specification
Specification of existing PF dump tapes
Specification of PFs to copy from existing dump tapes

All parameters are described according to function in Section 5.4.3. They are presented here in alphabetical order.

ADD=addlfn[=...] specifies up to 5 local files which hold a list of the PFs to be dumped from disk onto tape. If ADD appears alone, ADD=INPUT is assumed. (See PFDUMP Input Lists, Section 5.4.3, for format specifications.)

CY=xx allows a cycle number to be specified in conjunction with the PFN parameter, and is illegal unless the PFN parameter is also used. Legal forms are:

CY=xx where xx is the cycle number to be dumped $1 \leq xx \leq 63$.

CY=ANY — causes the first cycle encountered on disk to be dumped. If CY is omitted, CY=ANY is assumed.

CY=ALL — causes all cycles of that PFN to be dumped.

DROP=droplfn[=...]

specifies up to 5 local files which hold a list of the PFs to be skipped during the copying of the old dump tapes to the new dump tapes. All PFs not on this list will be copied onto the new dump tapes unless they already exist on the new dump tapes.

IX=indxlfn

specifies the file on which PFDUMP will write an index output file, which is a list of those PFs that were actually dumped on the new tapes. PFs are listed in the order that they occur on the new dump tapes, along with information about the tapes on file.

KEEP=keeplfn[=...]

specifies up to 5 local files which hold a list of PFs to be copied from the old dump tapes to the new dump tapes.

MT=vrn[=...]

specifies which tapes are to be written; this applies only to 7-track tapes. MT alone or in conjunction with the NT parameter is illegal.

NEWPN=nnnnnnn

specifies the PN to be written on the new tape labels. The value 'nnnnnn' may be a 6 or 7 digit problem number. If NEWPN is not specified, the PN under which the job is running is used. If NEWPN=0 is specified, any subsequent job will be allowed to rewrite the new tapes. Any non-zero value for NEWPN restricts the ability to rewrite the new tapes to the designated PN.

NT=vrn[=...]

specifies which tapes are to be written; this applies only to 9-track tapes. NT alone or in conjunction with the MT parameter is illegal.

O=outlfn

specifies the file upon which PFDUMP will echo all input lists (along with appropriate diagnostics) followed by a commentary on every PF that is processed. If O is not specified or O appears alone, O=OUTPUT is assumed.

OLDMT=vrn[=...]

specifies the VRNs of one or more existing 7-track PF dump tapes that are to be merged with the PF's dumped from disk. OLDMT alone is illegal. Up to 20 OLDMT and OLDNT requests are allowed.

OLDNT=vrn[=...]

specifies the VRNs of one or more existing 9-track PF dump tapes that are to be merged with the PF's dumped from disk. OLDNT alone is illegal. Up to 20 OLDMT and OLDNT requests are allowed.

ORDER

specifies that the disk PFs are to be dumped in the order in which they are specified in the ADD list. This is normally the case, but if PFDUMP must wait for a PF that is in use, the order might not be retained without the use of this parameter.

PFN=xx

specifies a single PF to be dumped. The PFN may be delimited '$', allowing the use of special characters within the PFN (in which case a '$' within the PFN must be represented by '$$').

U=unhdlfn

specifies the file on which PFDUMP will write an unheadered output file containing one line for each PF that it processes. The format and content is the same as the 'O' file without the echo of input lists. If U is not specified, no such file is generated. If U appears alone, U=UNHEAD is assumed.

WAIT

specifies that PFDUMP should wait for any PFs that are in use by another job. Use of the WAIT parameter does not ensure that the PFs will be dumped in the order specified in the ADD list. If 'WAIT' is not specified, PFDUMP will skip the file in use, noting the action in the output file.

Note: The PFDUMP control statement allows processing of continuation cards in batch mode. The parameter list may be separated after any comma and continued on the next card.

Examples: Further documentation of PFDUMP, including examples, can be found in Section 5.4.3.

7.14.5 PFLIST

PFLIST provides a printed report on the status of selected permanent files. Normally PFLIST obtains information only for the files charged against the user's ID dollar balance or, in the case of a PN manager, for all files charged against the PN dollar balance. Control statement options allow the user to select many other selection criteria as well.

PFLIST[,optional parameters].

PFLIST parameters can be divided into two functional categories:

Format control
File selection

The parameters are discussed according to function in Section 5.3. They are presented here in alphabetical order.

ACCESS=mm/dd/yy
List only the files that have been attached on or after the date specified. If ACCESS alone or ACCESS with no date is specified, the current date is assumed.

ALL
List all files satisfying the options specified, including those cataloged under different PNs. If ALL is the only parameter specified, all permanent files in the system will be listed.

ALTERED=mm/dd/yy
List only the files that have been altered on or after the date specified. If no date is specified, list only the files that have been altered since their last dump date.

ATTACH=noatt
List only the files whose attach count is less than or equal to that specified. If the ATTACH keyword appears alone, ATTACH=0 is assumed.

BACKUP=U
If the BACKUP keyword appears alone, list only the files which are not adequately backed up. Only verified dumps which are not obsolete are counted. If BACKUP=U is specified, unverified dumps will also count.

BATCH
Write labeled output files in a format suitable for the line printers (see Table 5-1). BATCH is assumed for batch jobs unless the TTY parameter is specified. This format may also be used for terminals with a wide (136 column) carriage.

DUMPED=mm/dd/yy
List those files dumped on or after the specified date. If no date is specified, the current date is assumed.

EXPIRED=mm/dd/yy
List only the files that expire on or before the date specified. If no date is specified, the current date is assumed.

FDO=fdolfn
Designates the file on which PFLIST will write a full detail output file. All information (except dump information) about a particular PF will be listed on one very long (400 characters maximum) line. This file is intended for user programs to process and extract the desired information; it is not normally generated.

FULL

List both purged and active files.

FULLDMP

Lists all files for which BC‖0 (i.e. all files which are backed up by the Computer Laboratory) is specified.

FULLVRN

List full dump history for each file. A line giving the dump date/time and the dump tape VRN is printed for each time the file was dumped.

ID=id[=...]

List only the files cataloged with one of the IDs specified. This does not refer to the ID of the job card, but to the 'ID=id' parameter of the CATALOG statement.

LASTACC=mm/dd/yy

List only the files that were last accessed on or before the date specified. If no date is specified, the current date is assumed.

LASTALT=mm/dd/yy

List only the files that were last altered on or before the date specified. If no date is specified, the current date is assumed.

LASTCAT

Print the date on which the file was last cataloged in place of the creation date and the time when last cataloged in place of the last alteration time.

LISTPN

Print the problem number (PN) of the PF owner for each file in place of the cost.

MT=vrn[=...]

Search the specified 7-track PF dump tape set rather than the system permanent file directory.

NT=vrn[=...]

Search the specified 9-track PF dump tape set rather than the system permanent file directory.

O=outlfn
O=0

Specifies the file on which PFLIST is to write a labeled report (i.e., containing page headers). The default file name is OUTPUT for batch jobs and TTYTTY for interactive jobs (which means that output will not be listed at the terminal until PFLIST has ended). If the O parameter appears alone, O=ZZZZOT is assumed for interactive jobs (where ZZZZOT is connected). O=0 directs that labeled output not be generated.

PFN=pfn

List only the specified PF. This parameter will accept '$' delimiters when special characters are included in the permanent file name. A '$' within the name must be presented as '$$'.

PN=pn[=...]

List only those files that have been cataloged under the PNs specified. If the PN keyword appears alone, the PN of the user is assumed. 'PN' is the default option for problem number managers unless the ALL, PNDEPT, or an explicit PN or PNORD parameter is used.

PNDEPT=dept

List only the files cataloged under department code 'dept', where 'dept' is the first two characters of the problem number. If the PNDEPT keyword appears alone, the department code of the user running the job is assumed.

PNORD=pnord[=...]

List only those files that have been cataloged under the PN ordinals specified. The PN ordinal uniquely identifies each PN subaccount (user ID) in the Authorization File. If the PNORD keyword appears alone, the PN ordinal of the user who is running the job is assumed. This is the default option for non-PN managers unless the ALL, PN-DEPT, PN, or an explicit PNORD parameter is used. The PNORD can be obtained with the AUTHORF utility: 'AUTHORF,DISPLAY,PNORD.'

PREFIX=prfx

List the permanent files whose names begin with the characters specified. A prefix of up to 40 characters may be specified. This parameter will accept '$' delimiters when special characters are to be in the prefix. A '$' within the prefix must be represented as '$$'.

PURGED

List only the files that have been purged (since the last initial deadstart or within approximately three weeks; see Section 1.4.7).

SIZE=pru

List only the files that are larger than or equal to the number of PRUs specified.

SORT=sortkey

Sort the output (on the file specified by O=outlfn or U=unhdlfn) as indicated by key, where 'sortkey' is one of the following keywords:

ALPHA	sort by permanent file name
ACCOUNT	sort by PN and PN ordinal within the PN
PN	sort by PN
SIZE	sort by size (smallest first)
LASTACC	sort by last access (most recent first)
0	no sort

SORT=ALPHA is assumed if the SORT keyword appears alone. SORT=0 is assumed if the SORT option is omitted.

SOURCE=srce

List only the files that were cataloged from the source specified (see Appendix E). If the keyword SOURCE is used alone, SOURCE=B is assumed.

TTY

Write labeled output files in a format suitable for a 72-column terminal (see Table 5-2). TTY is assumed for interactive jobs, unless the BATCH parameter is specified.

U=unhdlfn

Specifies the file on which PFLIST is to write an unlabeled report. This file will always be in BATCH format. This output is more suitable for input into another program, such as PFDUMP or PN-PURGE. If the U parameter appears alone, U=UNHEAD is assumed.

VRN

List the VRN(s) of the tape(s) on which each permanent file was last dumped. This information is printed in place of the last alteration, last access, and number of attaches data.

Note: The PFLIST control statement allows processing of continuation cards in batch mode. The parameter list may be separated after any comma and continued on the next card.

Batch/Interactive Differences:

The report format is condensed if the TTY option is selected (default for interactive users). The formats of the TTY and BATCH options are described in Section 5.3.

Cautions:

1. The file specified by the 'U=unhdlfn' parameter is returned before each PFLIST statement.

2. The ALL option is very expensive and generates several hundred pages of output, unless used with ID, PREFIX, or other restricting parameter.

3. Information for purged files generally is retained only as long as there is a backup copy of the file on magnetic tape (see Section 5.4). Information for purged files may be lost prematurely in the event of a system failure that requires a reload of all permanent files (an initial deadstart).

Examples: Examples and sample PFLIST reports may be found in Section 5.3.

7.14.6 PFLOAD

The PFLOAD control statement causes one or more permanent files to be reloaded from a dump tape. The file(s) is reloaded with the cataloging information intact.

PFLOAD[,optional parameters].

The parameters are divided into four functional groups:

Tape specification
PF specification
Recataloging information
Output file specification

The parameters are discussed according to function in Section 5.4.4. They are presented here in alphabetical order.

ALL — reload all PFs on the specified tapes.

CY=opt — allows a cycle number to be specified in conjunction with the PFN parameter. This parameter is illegal unless the PFN parameter is also used. Legal forms are:

- CY=cy — where cy is the cycle number to be reloaded. 1≤cy≤63.
- CY=ANY — causes the first cycle encountered on the tape set to be reloaded. If CY is omitted, CY=ANY is assumed.
- CY=ALL — causes all cycles of that pfn to be reloaded.

DUP=opt — specifies the action to be taken when PFLOAD attempts to recatalog a PF and finds that the pfn, or pfn and cycle number, are already in use by another disk PF. Legal forms are:

- DUP=IGNORE — causes PFLOAD to skip this PF (noting this action in the output file) and continue on to the next PF.
- DUP=NEWNAME — causes PFLOAD to create a new pfn, cataloging the PF under the new name. DUP=NEWNAME is assumed if DUP is not specified.

I=inlfn[=...] specifies up to 5 names of local files which hold a list of the PFs to be reloaded.

MT alone, indicates that the tapes whose VRN's are supplied in the input list without a tape type (7- or 9-track) specification are 7-track; illegal with the NT parameter.

MT=vrn[=...] specifies a global tape set to be loaded; this applies only to 7-track tapes.

NT alone, indicates that the tapes whose VRN's are supplied in the input list without a tape type (7- or 9-track) specification are 9-track; illegal with the MT parameter.

NT=vrn[=...] specifies a global tape set to be loaded; this applies only to 9-track tapes.

O=outlfn specifies the file upon which PFLOAD will echo the control statements, all input cards, and then list all the results of the reload attempts. If O is not specified, O=OUTPUT is assumed.

PFN=pfn specifies a single PF to be reloaded. The pfn may be delimited by '$', allowing the use of special characters with the PFN (in which case a '$' within the PFN must be represented by '$$').

RP=opt specifies the retention period of the reloaded PF's in days. Legal forms are:

RP=rp where 0≤rp≤999. If RP is omitted, RP=15 is assumed.

RP=SAME uses the RP value assigned to each PF before it was dumped.

U=unhdlfn specifies the file upon which PFLOAD will write an unheadered list of the results of the reload attempts. U alone implies U=UNHEAD.

Note: IF U is omitted, no unheadered output is generated.

Note: The PFLOAD control statement allows processing of continuation cards in batch mode. The parameter list may be separated after any comma and continued on the next card.

Examples: See Section 5.4.4 for complete documentation and examples of PFLOAD use.

7.14.7 PNPURGE

PNPURGE allows the user to purge any permanent file charged to his/her account without having to attach the file or to specify any of its passwords. The PN manager can purge any file cataloged under the PN. Other users can purge only the files cataloged under their own IDs.

```
PNPURGE[,optional parameters].
```

CY=cy cycle number of the file to be purged. If omitted, the highest numbered cycle is purged.

DPFN=dpfnd specifies the permanent file, where 'pfn' is the permanent file name and 'd' is a delimiter character specified by the user. The delimiter character must appear immediately after the equal sign; it may be any character (including blank) that does not appear in the pfn. All characters between the first and second occurrence of the delimiter—including blanks, commas, periods and parentheses—are treated as part of the name.

I = inlfn — the name of a file containing a list of the files to be purged. The default file name is INPUT. Each line of this file should conform to the following format:

columns 3-42 permanent file name (left-justified)
columns 55-56 cycle number

This is the same format created by the PFLIST unheadered output option (see Section 5.3).

PFN = pfn — permanent file name of the file to be purged.

Cautions:

1. The errors "FILE NOT ON SYSTEM" and "CYCLE REFERENCED DOES NOT EXIST" are non-fatal. Other permanent file errors cause an abort only after PNPURGE has finished processing the input list.

2. PNPURGE cannot purge a file that is currently attached.

Example:

PNPURGE is often useful in interactive mode. Connect INPUT, turn on PROMPT, and remember to type two spaces before the permanent file name. To exit, enter *EOR or *EOF. (User type-ins are shaded.)

```
CONNECT,INPUT. PROMPT. PNPURGE.
*  JALAUTHORFEWF
*  JALAUTHORFLGO
*  JALAUTHORFPL
**EOR
```

7.14.8 PURGE

The PURGE control statement changes an attached permanent file to a temporary file.

PURGE,lfn.

lfn — the local file name of the permanent file to be purged.

To execute the PURGE statement, the permanent file must be attached with control permission. If a file is to be purged and then recataloged, it must be attached with all permissions, i.e., all associated passwords must be specified.

The local file is still available to the job after the PURGE. However, even after the file is purged, the permissions under which it was attached are still in effect; for example, if read permission was not granted on the ATTACH statement, it will not be granted after the PURGE. If the purged file is displayed by FILES (see Section 7.4.3), it will still be considered as permanent unless all permissions were granted on the ATTACH statement.

Examples: See Section 5.2.3 for further discussion and examples.

7.15 Libraries

The utilities described in this section provide facilities for the maintenance of programs, subprograms, and data files on libraries, either on disk or tape.

7.15.1 APLIB

APLIB is used to maintain a library of user files; these files may contain any type of information in either binary or coded format. APLIB maintains the files on a 7-track magnetic tape, permanent file, or both. APLIB uses a set of directives to add, delete, retrieve, and replace files on the library.

APLIB is called by the following control statement:

```
APLIB,lib[,opts].
```

lib the library name lib is of the form

<label><mode><name>.

where 'label' is

U	no labels
L	standard SCOPE labels
omitted	standard SCOPE labels

'mode' is

MT	permanent file with tape backup
TT	tape-only mode
PF	permanent file only

'name' is the tape VRN for MT and TT modes (except SYSxxx, VIMxxx, or UPxxxx) or any 1-6 alphanumeric characters for PF mode.

Examples:

MT1234
UTT625
LMTLY017

opts a string of options. Options may be continued on successive cards by omitting the terminator from all but the last card. (The terminator is a period or closing parenthesis.) The options are described below.

Update Options

The characters I, N, T, U, Z designate update or creation operations. If one is used, it must be the first option listed; retrieval options (see below) may follow it. The update codes mean:

I* update only individual permanent files
N* create a new library
T* make disk-only modifications
U* update both tape and disk
Z* same as U*, but also changes unlabeled tape to labeled

If update directives are to be read from a file other than INPUT, the name of the file immediately succeeds the option code. For example, 'APLIB,MT2530,U*A.' causes directives to be read from file A.

Retrieval Options

The remaining options indicate retrieval operations. The mnemonics 'libfn' and 'lfn' mean 'library file name' and 'local file name' respectively. The library file name is the identifier given to the file within the library. The local file name is the name of the file to which it is copied when it is retrieved. Both must be 1-7 alphanumeric characters, and the first character must be alphabetic.

libfn — library file libfn will be copied to file LGO behind any information already present. LGO is not rewound; an end-of-partition (EOP) will be written at the end and then backspaced over.

*libfn — library file libfn will be copied to a local file of the same name. The local file is rewound before and after the copy.

A*libfn — similar to *libfn, except that the local file is not rewound before or after the copy; an end-of-partition (EOP) is written at the end and then backspaced over.

R*libfn=lfn — similar to *libfn, except that lfn is the local file name used for the copy, e.g., library file libfn is copied to local file lfn.

RA*libfn=lfn — a combination of the A*libfn and R*libfn=lfn forms. Library file libfn is copied to local file lfn, and lfn is not rewound before or after.

P*libfn — if an individual permanent file copy of libfn exists, it is attached as local file libfn. If not, the file is retrieved from the library, copied to local file libfn, and cataloged with a retention period of one day. The permanent file name of a file retrieved by 'APLIB,MT123,P*SVFILE.' would be MT123SVFILE.

PR*libfn=lfn — similar to the P*libfn option, except that lfn is used as the local file name for the attach or copy.

C*libfn
CR*libfn=lfn — similar to the P*libfn and PR*libfn=lfn options, except that if a permanent file is created, it is given a retention period of zero.

G*libfn
GR*libfn=lfn — similar to the P*libfn and PR*libfn=lfn options in that they will attach the individual permanent file if it exists. But if it does not exist, this option merely copies the requested file from the library to a local file and does not catalog the local file.

L*libfn — library file libfn will be copied to OUTPUT. The file is assumed to have appropriate carriage controls in column one. OUTPUT is not rewound before or after the copy.

L* — this will list the library directory on file OUTPUT.

Update Directives

There are seven update directives. When an update control statement option is used, APLIB will read update directives from file INPUT, (or an alternate file as directed by the control statement). Each directive must begin in Column 1 of a separate card or line. Commas and blanks are used to delimit fields within the directives.

ADD,libfn,lfn,NR,MF,PF,RS,EF.

This directive adds file libfn to the library. All codes following libfn are optional.

libfn the library file name given to the file. If lfn is omitted, this is also the local file from which it is read.

lfn the local file name from which libfn is copied.

NR specifies that the source file is not to be rewound before or after copying it to the library.

MF specifies that the source file is to be copied to end-of-information rather than end-of-partition. **Caution:** This parameter has the effect of insuring that all the data is copied to the library regardless of an embedded end-of-partition.

PF specifies that the libfn is to be cataloged for subsequent use by G*, P*, and C* retrievals.

RS specifies that the source file is to be returned (CLOSE,UNLOAD) after it has been copied.

EF specifies that a terminating end-of-partition (EOP) is to be written when this file is later retrieved, regardless of whether or not the original copy contains one.

The statement 'ADD,libfn.' means the same as 'ADD,libfn,NR,MF,RS,EF.'

If the source file (libfn or lfn) is empty or does not exist, an informative message is given. If libfn is already defined in the library, a 'D' is prefixed to the name to make it unique.

DELETE,libfn.

The library file libfn is permanently removed from the library.

REPLACE,libfn,lfn,NR,MF,PF,RS,EF.

This performs a DELETE followed by an ADD on libfn. If the source file does not exist, the deleted file is restored. The parameters NR, MF, PF, RS, and EF have the same meanings as on the ADD directive.

RENAME,oldlibfn,newlibfn.

The library file name oldlibfn is changed to newlibfn.

SAVEPF,libfn,MF,PF,EF.

This directive causes the individual permanent file copy of libfn to be attached and used to REPLACE (or ADD) the library copy of libfn. The permanent file will then be purged unless PF is specified. If libfn is specified as *ALL, APLIB will attempt a SAVEPF for every file in the library which has not already been deleted or replaced in this update. The parameters MF, PF, and EF have the same meanings as on the ADD directive.

```
NEWPN=pn
```

This card puts a new problem number (PN) in the SCOPE volume header label recorded on the magnetic tape. If the tape is currently unlabeled, a Z* update is performed. The 'pn' field must be in columns 7-13.

SC Code

The security code card specifies a password to protect the library from being changed by unauthorized persons. This card can be placed anywhere among the update directives to initially specify the code. Thereafter any update directive set must begin with an identical SC card. A second SC card in the set will change or clear the security code. 'Code' may be any 1-10 characters in columns 4-13. If this field is blank, the security code is cleared.

Example 1:

```
PNC
job card
COPYCR,INPUT,DATA.
UPDATE,N.
FTN,I=COMPILE.
APLIB,TT123,N*.
LGO,DATA.
7/8/9
  data                          Library creation run retaining individual PF's.
  .
  .
  .
7/8/9
*DECK PROG
       PROGRAM PROG (INPUT,OUTPUT)
       .
       .
       .
7/8/9
ADD,PROGPL,NEWPL,PF,RS
ADD,PROGLGO,LGO,PF
ADD,DATA,PF
6/7/8/9
```

Example 2:

```
PNC
job card
APLIB,TT123,PR*PROGPL=OLDPL,P*PROGLGO,P*DATA.
UPDATE,N.                               Retrieval and update of individual PF's.
FTN,I=COMPILE.
REWIND,LGO.
COPYL,PROGLGO,LGO,PROG.
PROG,DATA.
APLIB,TT123,I*.
7/8/9
*IDENT MODS
.
.
.
7/8/9
REPLACE,PROGPL,NEWPL,PF.
REPLACE,PROGLGO,PROG,PF.
6/7/8/9
```

Example 3:

```
PNC
job card
APLIB,TT123,U*.                         Update of tape library.
7/8/9
SAVEPF,*ALL.
6/7/8/9
```

Note: APLIB sets RERUN to OFF in case problems occur when rewriting an APLIB tape.

7.15.2 ARCHIVE

ARCHIVE is a utility to retrieve programs from the MSU archive library tapes. ARCHIVE is called by a control statement of the form:

```
HAL,ARCHIVE,lfn[,C][,dname[,...]].
```

lfn — is the local file name onto which archive programs are to be copied. If lfn is LIST, the requested archive programs are listed.

C — controls interactive control statement processing after the ARCHIVE statement (see below).

dname — any archive program name. Current listings of archive programs are available in the User Information Center, Room 313 Computer Center.

From batch, ARCHIVE creates a control statement section to retrieve all of the specified programs from the archive library tapes. The remaining user control statements are copied to the back of the archive control statements. An EXEC is then performed.**Caution:** Because of the EXEC done by ARCHIVE, ARCHIVE should not be called more than once in one batch job.

Since ARCHIVE uses magnetic tapes, it can not be used directly from an interactive terminal; ARCHIVE creates a complete batch job and disposes it to the input queue. Users will be asked to supply information which ARCHIVE uses to create a job card for the disposed file. If the C parameter follows lfn on the ARCHIVE control statement, any statements following on the same line as the ARCHIVE statement will be copied to the back of the ARCHIVE control statements before the file is disposed to INPUT.

Note: The MT1 parameter must be included on the job card when using ARCHIVE.

7.15.3 COPYCL

COPYCL is used to create, maintain, and update COBOL source libraries for use with the COPY and INCLUDE statements of the COBOL language.

The control statement is of the form:

COPYCL[,optional parameters].

COMPILE=complfn
COMPILE=0 If complfn is specified, complfn is written as a COBOL source library. If COMPILE is specified alone, the COBOL source library will be written on the file COLIB. If this parameter is omitted, or if COMPILE=0 is given, no source library is written. COMPILE may be abbreviated C.

INPUT=inlfn If INPUT=inlfn is specified, the file inlfn will be searched for corrections. If INPUT is specified alone or omitted, the file INPUT will be searched for corrections. INPUT may be abbreviated I.

LIST=listlfn If LIST=listlfn is specified, output will be listed on the file listlfn. If LIST is
LIST=0 given alone, output will be listed on OUTPUT. If LIST=0 is specified, no list is generated. LIST may be abbreviated L.

NPL=nplfn If NPL=nplfn is specified, the new program library will be on nplfn. If NPL is
NPL=0 specified alone, the new program library will be on NPL. If NPL=0 is given, no new program library is generated.

OPL=oplfn If OPL=oplfn is given, the old program library will be found on oplfn. If OPL
OPL=0 is given alone, the old program library is on the file OPL. If OPL=0 is given, no old program library is assumed.

For complete documentation on input directives, and examples, see Section II-12 of the *COBOL Reference Manual.*

7.15.4 HAL

HAL is the system utility through which the user accesses the subprograms, program packages, and data files of the HUSTLER Auxiliary Library. HAL provides similar means for creating, updating, and referencing user auxiliary libraries. A special feature of these libraries is the ability to create and retrieve documentation for user auxiliary library entries in the same way that documentation is maintained and retrieved for the HUSTLER Auxiliary Library, via the program HELP.

An entry in the HUSTLER Auxiliary Library may be one of four types:

1. subprogram (binary relocatable form)
2. main program or program system (any loadable stand-alone module)
3. data file (neither of the above)
4. documentation (information to be retrieved by the HELP utility)

The entire HAL system is described in the *HAL Reference Manual*. For convenient reference, several forms of the HAL control statement are described below.

Subprogram Retrieval

Before loading a FORTRAN program, the user may request HAL to satisfy external references on the binary load-and-go file (e.g., LGO), or he/she may have HAL copy specific routines to a local file (normally LGO).

The HAL control statement for the two forms of subprogram retrieval are:

```
HAL.
```

This statement causes the loader to use the HUSTLER Auxiliary Library for satisfying externals. This statement must immediately precede any load sequence to be affected.

```
HAL,lfn=sub[,...].
```

This statement copies named subprograms to file lfn without rewinding lfn before or after the copy. The file lfn cannot be the reserved word B.

Example 1:

```
PNC
job card
PW=password
FTN.
HAL.        or      HAL,LGO=RANDU.
LGO.
7/8/9
FORTRAN program
.
.
.
CALL RANDU(IX,IY,YFL)
.
.
.
7/8/9
data
6/7/8/9
```

Substituting 'HAL,LGO=RANDU.' for 'HAL.' performs the same task but subroutine RANDU will be copied to LGO; it will not be copied using the 'HAL.' form.

Main Program (Overlay) Retrieval

Program packages or main programs do not require a user-supplied program for their use. These programs can be loaded and executed with a single control statement. There are two forms of the HAL control statement for retrieving main programs.

```
HAL,pname[,optional parameters].
```

This form retrieves, loads and executes program pname. Optional parameters are passed to program pname. A local file copy of pname is made; this copy is subject to the cautions described below.

```
HAL,*pname.
```

This form retrieves program pname but does not load or execute it. See the cautions described below.

Caution: Care should be taken in using the 'HAL,*pname.' form. "Load-only" overlays can only be accessed with 'HAL,pname.' Programs retrieved with 'HAL,pname.' or 'HAL,*pname.' and then used later in the job stream may not have enough field length to load. In addition, some programs are left on a local file other than lfn pname. For example:

```
HAL,STAT4.
```

leaves program STAT4 on local file STAT instead of STAT4.

Example 2:

```
PNC
job card
PW=password
REQUEST,TAPE1,VRN=1025.
HAL,FILEDMP,I=TAPE1.
RETURN,TAPE1.
REQUEST,TAPE2,VRN=1183,S,Z.
FILEDMP,I=TAPE2,BCD.
6/7/8/9
```

In this example, HAL copies main program FILEDMP to local file FILEDMP and then loads and executes it. The second time FILEDMP is loaded and executed from its local file copy.

Data File Retrieval

Any entry on the HUSTLER Auxiliary Library which is not classified as a subprogram or main program is considered a data file. Data files may be referenced in three ways:

```
HAL,dname.
```

Copies data file dname to lfn dname (see Caution below). The file dname is not rewound.

```
HAL,*dname.
```

Copies data file dname to lfn dname (see Caution below). The file dname is rewound before and after the copy.

```
HAL,E*dname.
```

Copies data file dname to lfn HALEXEC and executes control statements on file HALEXEC. An error results if dname is not a data file.

Caution: For methods 1 and 2 above, some data files are not copied to dname, but to another local file that was specified at the time the entry was created. 'HAL,INDEX.' can be executed to determine the local file name.

Example 3:

```
PNC
job card
PW=password
FTN.
HAL,E*TEKTRONIX.     executes control statements on TEKTRONIX.
LGO.
7/8/9
program which calls TEKTRONIX routines
6/7/8/9
```

HAL performs a 'CCEXEC' (see Section 7.1.1); that is, all control statements (in this case, 'LGO.') after the 'HAL,E*TEKTRONIX.' statement will be executed.

Example 4:

```
PNC
job card
PW=password
HAL,*WORDS.          retrieves data file WORDS; WORDS is then rewound.
FTN.
LGO,WORDS.
7/8/9
program which uses WORDS as data input
6/7/8/9
```

More examples may be found in the *HAL Reference Manual*, Chapter 2.

7.15.5 LIBEDIT

The LIBEDIT utility allows a user to define a group of central processor routines or overlays as a library. A LIBEDIT user library can be modified by the addition, deletion, or replacement of routines; and statistics about library contents can be listed.

A user library may contain compiled programs and subroutines, loader-generated overlays, and assembled COMPASS system texts. The following may not be made a part of a user library: source language programs, peripheral processor programs, binary output from SEGLOAD, and OVERLAY directives.

The LIBEDIT utility is called by the LIBEDIT control statement. A parameter on this statement specifies the file that contains LIBEDIT directives. These directives provide details for creating or manipulating the user library. Note: Parameters on LIBEDIT directives must be enclosed in parentheses.

The control statement is:

```
LIBEDIT[,I=dirlfn][,L=listlfn].
```

I=dirlfn specifies the name of the local file containing the directive section. If I is given alone or if I is omitted, the file INPUT is assumed.

L=listlfn specifies the name of the local file to receive listable output. If L is given alone or if L is omitted, the file OUTPUT is assumed.

When the LIBEDIT control statement is encountered during job processing, LIBEDIT will access the next unprocessed record on the INPUT file, unless the I parameter names another source of directives.

Example:

The following example assembles a COMPASS program, compiles a FORTRAN program, and adds them to an existing library:

```
PNC
job card.
PW=password
COMPASS.
FTN.
ATTACH,ALIB,USERLIBRARY.
LIBEDIT.
7/8/9
  :                         COMPASS program to be assembled
7/8/9
  :                         FORTRAN Extended program to be compiled
7/8/9
LIBRARY(ALIB,OLD)
REWIND(LGO)
ADD(*,LGO)
ENDRUN.                     Directives instructing LIBEDIT to add programs to user
                            library ALIB from LGO file.
6/7/8/9
```

The use of LIBEDIT is documented in *User's Guide Supplement: LIBEDIT—Cyber Loader Libraries*.

7.15.6 RANLIB

RANLIB is a facility which allows the user to create and modify a library of binary program decks. Groups of binary decks can be defined and referenced with a single name. Decks can also be grouped into OVERLAYS. When this is done, RANLIB will automatically generate any necessary OVERLAY cards. RANLIB was designed especially to maintain large overlayed programs, but can be used with non-overlayed programs as well.

```
RANLIB[,optional parameters].
```

The parameters are selected from the following in any order. All parameters except D, SL, and HELP can be followed by an optional local file name.

BI=binlfn

binary input file. If BI=binlfn is specified, the binary input file will be the file binlfn. The default file is LGO.

BO=binolfn

binary output file. If BO=binolfn is given, the decks specified on *BOUT cards (see below) are written on the file binolfn. The default file is BOUT.

D

debug parameter. RANLIB prints a warning message if it cannot find a deck mentioned in a *BOUT card. If D is omitted, RANLIB aborts if it cannot find a deck mentioned in a *BOUT card.

HELP

help message. A two page help message is written on the file specified by the O parameter. If omitted, no help message is printed.

I=inlfn

input file. RANLIB directives are read from file inlfn. The batch default is INPUT; the interactive default is the connected file ZZZZIN.

LBI=lstblfn

list binary input. If LBI=lstblfn is given, the index to the BI file is listed on the file lstblfn. If LBI is given alone, the index to the BI file is listed on the file specified by the O parameter. If LBI is omitted, the index is not listed.

LD=deflfn

list DEFs. If LD=deflfn is given, the DEF table is listed on the file deflfn. If LD is specified alone, the DEF tab is listed on the file given with the O parameter. If LD is omitted, the DEF table is not listed.

LI=indxlfn

list index. The index is listed on the file indxlfn if specified. If indxlfn is not specified, the index will be listed on the file given with the O parameter. If LI is omitted, the index is not listed.

LID or LDI=indelfn

list both DEF table and index. If indelfn is given with LID or LDI, both index and DEF table are listed on the file indelfn. If indelfn is not specified, the index and DEF table are listed on the file given with O. If this parameter is omitted, no change is made in the LI or LD settings. Note: LID or LDI overrides LI or LD.

N=nliblfn

new library file. If N=nliblfn is given, a new library is created on the file nliblfn and all changes are made to it. The old library is unchanged. If N is given with no nliblfn, changes are made directly to the old library. If N is omitted, no changes can be made to the library.

O=outlfn

output file. Listable output is listed on the file outlfn. The default batch output file is OUTPUT; the interactive default is the connected file ZZZZOT.

P=oliblfn

old library file. If P=oliblfn is specified, the library file is on the file oliblfn. The default file is LIBRARY.

SL

short list. If the DEF table is listed, DEFs are expanded only one level. If omitted, DEFs are expanded completely.

RANLIB Directives

RANLIB directives are divided into four classes. All directives in a given class must appear before any directives in a following class. Any directive can be continued by putting a comma at the end of one card and a blank in column 1 of the next card. In all examples, the following definitions apply:

A 'deck' is a binary deck name.
A 'name' is a group of progs defined by a *DEF directive.
A 'prog' is either a name or a deck.

Class 1 Directives:

```
*ADD lfn[/NR][=deck[,...]]
```

*ADD adds or replaces decks from the file lfn to the library. If a string of decks is specified, only those are added, otherwise all of the decks on the file lfn will be added. If the /NR is used, file lfn will not be rewound before reading.

```
*NEWADD lfn[/NR][=deck[,...]]
```

*NEWADD works like *ADD except that it will not replace. If a deck on file lfn is already on the library, then it is skipped.

```
*DEL deck[,...]
```

*DEL deletes the specified decks from the library.

Class 2 Directives:

```
*DEF name = [*([lfn,]n,m),]prog[,...]
```

*DEF defines 'name' to reference a group of progs, and an optional overlay card, represented by the *. (The word OVERLAY may replace the * in this directive.) A use of this name in a *BOUT or *EXTRACT card will cause all of the progs defined in the DEF to be output.

If RANLIB is used to generate overlays, the overlay name and levels may be specified in the *DEF directive, 'lfn' referring to the overlay name and 'n' and 'm' referring to the overlay levels.

```
*DELDEF name[,...]
```

*DELDEF deletes the given name(s) (specified in a previous *DEF directive) from the library.

Class 3 Directives:

```
*LISTDEF name[,...]
```

*LISTDEF lists the specified names on the O file; if SL is specified on the control statement, then the names are expanded only one level deep; if not, names are fully expanded.

Class 4 Directives:

```
*BOUT prog[,...]
```

*BOUT expands names into OVERLAY cards and binary decks for the loader and writes them on the BO file. Decks are taken from the BI file if possible.

```
*EXTRACT lfn,prog[,...]
```

*EXTRACT creates a sublibrary on file lfn of those progs specified. Only those decks mentioned that are on the library will be written; the BI file is ignored.

Interactive Usage of RANLIB

1. In interactive mode, if the default input file of ZZZZIN is used, RANLIB will prompt with ?? for directives.

2. Pressing the abort key (normally ESCAPE) while RANLIB is processing may destroy the library.

All other RANLIB features are identical in batch and interactive usage.

Example 1: Library creation

```
PNC
job card
PW=password
FTN.
RANLIB,N,LDI,SL.                          Add decks to LIBRARY and generate
                                          BOUT file for loader.
CATALOG,LIBRARY,MYRANLIBLIBRARY.
LOAD,BOUT.                                Generate OVERLAYs on SETUP.
NOGO.
CATALOG,SETUP,MYSETUPPROGRAM.
7/8/9
FORTRAN program and subroutines
7/8/9
*ADD LGO.                                 Create library consisting of the decks on
                                          LGO.
*DEF OV0=*(SETUP,0,0),EDITOR,MAIN         Define decks to belong to various
                                          overlays.
*DEF OV1=*(1,0),PARSE,GETWORD
*DEF OV2=*(2,0),SUB1,SUB2
*DEF OV3=*(3,0),ILEDIT,ERROR
*DEF ALL=OV0,OV1,OV2,OV3
*DEF MAIN=READIN,WRITOT,ADD
*BOUT ALL
6/7/8/9
```

Example 2: BI File use

```
PNC
job card
PW=password
FTN.
ATTACH,LIBRARY,MYRANLIBLIBRARY.
RANLIB,LBI.                                LGO has a new version of ILEDIT and
                                           USE
7/8/9
FORTRAN subroutines ILEDIT and USE
7/8/9
*BOUT ALL                                  A binary file with the new decks in the
                                           proper OVERLAYS is on BOUT.
6/7/8/9
```

Example 3: Library modification

```
.
.
.
ATTACH,LIBRARY,MYRANLIBLIBRARY.
RANLIB,N=NEWLIB.
CATALOG,NEWLIB,MYNEWRANLIBLIBRARY.
PURGE,LIBRARY.
7/8/9
*NEWADD FILE1
*ADD FILE2=ILEDIT,USE
*DEF OV4=*(4,0)STRING,MACROS               Add new definition
*DEF ALL=OV0,OV1,OV2,OV3,OV4               Replace old definition of ALL
6/7/8/9
```

7.15.7 UPDATE

The UPDATE utility provides a means for creating, correcting, and referencing libraries of source decks. These decks may be source cards for compilation and assembly runs, data cards, or any other type of text. The program library created by UPDATE, referred to as a PL, stores these card images in a special, compressed format designed to facilitate updates.

UPDATE is controlled by an extensive set of control statement options and directives that allow the user to insert, delete (deactivate), and restore (reactivate) card images within the library and to perform many other operations helpful in maintaining a library of this type. For a creation run, the primary input to UPDATE is a file of text cards which are to be made into a PL. For a correction run, the input file contains editing directives interspersed with text cards that are to replace or to be inserted between existing lines of the PL. The primary output of an UPDATE run is a **compile file**, which contains selected decks in standard coded form. This file may be printed, punched, or used as input to a compiler, assembler, or other type of program.

Full documentation for UPDATE is available in the CDC *UPDATE Reference Manual* (Publication No. 60449900). The control statement is described below.

```
UPDATE[,optional parameters].
```

A

sequential-to-random copy. UPDATE copies a sequential old program library to a random new program library. The only other control statement options that can be used are those specifying files, R, * and /. No input is read.

B

random-to-sequential copy. UPDATE copies a random old program library to a sequential new program library. The only other control statement options that can be used are those specifying files, R, * and /. No input is read.

C=complfn
C=PUNCH
C=0

compile file output. If C=complfn is specified, UPDATE writes compile output decks on the named file. The default file is COMPILE. If C=PUNCH is given, UPDATE writes compile output decks on file PUNCH. This option also causes the 8 and D parameters to be selected. If C=0 is specified, no compile output is generated.

D

data width. Compile output has 80 columns for data. If omitted, compile output has 72 columns for data.

E

edit; provides a means of cleaning up old program libraries. UPDATE rearranges the directory to reflect the actual order of decks on the program library. If omitted, the old program library is not edited. During editing, UPDATE purges any cards other than YANK, SELYANK, and YANKDECK from the YANK$$$ deck. (See Section 3.2.1 of the *UPDATE Reference Manual*.

F

full update. Source and compile files, if specified, contain all active decks in old program library sequence. The contents of the new program library are the same as if F were not specified.

If Q is not specified, F omitted is the normal (selective) UPDATE mode. All regular decks and common decks are processed. The new program library, if specified, contains all regular and common decks, after any corrections have been made, in the sequence in which they occur on the old program library. The source file, if specified, contains all decks corrected during this UPDATE run and all decks specified on COMPILE directives. A deck that calls a corrected common deck is also considered to be corrected unless the common deck is a NOPROP deck or unless the deck precedes the common deck.

G=pmodlfn

generate separate PULLMOD output file. Output from PULLMOD cards is written on the file pmodlfn. Any rewind option applying to the source file also applies to this file. OUTPUT is not a valid file for this option. If this parameter is omitted, output from PULLMOD cards is appended to source file defined by the S or T parameter or to the file SOURCE.

I=inlfn

The input comprises the next section of file inlfn. The default is INPUT.

K=complfn

COMPILE card sequence (takes precedence over C parameter). If K=complfn is specified, compile output decks are written on file complfn in COMPILE directive sequence. If K is given alone, compile output decks are written on file COMPILE in COMPILE directive sequence, that is, the order in which deck names are encountered on *COMPILE directives. If K is omitted, the output is determined by the C option.

L = options list options. The options are selected and listed below:

A Lists the following:

1. Known DECK names
2. Known IDENT names
3. COMDECK directives that were processed (subset of 1)
4. Decks written on the compile file
5. Known definitions (see *DEFINE directive)

F All selections other than 0.

0 Suppresses all UPDATE listing. If the digit 0 is included in the string of other list selections, the string is equivalent to L=0.

1 Lists cards in error and the associated error messages. The flag *ERROR* is appended to the left and right of each card in error.

2 Lists all active UPDATE directives encountered either on input or on the old program library.

3 Lists all cards that changed status during this UPDATE. This listing consists of the name of the deck to which the card belongs, the card image, card identifier with sequence number, and a key as shown below:

Key	Meaning
I	Card was introduced
A	Inactive card was reactivated
D	Active card was deactivated
P	Card was purged
SEQ	Card was resequenced

If a currently active card is purged, the word ACTIVE appears to the right of the P.

4 Lists all non-UPDATE directives encountered in the input stream.

Decks inserted by ADDFILE are not listed if list option 4 is selected by default but they are listed if list option 4 is explicitly selected.

Option 4 may be turned on by a LIST card and off by a NOLIST card (see *UPDATE Reference Manual* for a description of directives).

5 All active compile file directives.

6 Number of active and inactive cards by deck name and correction set identifier.

7 All active cards.

8 All inactive cards.

9 Correction history of all cards listed as a result of list options 5, 7, and 8.

List options 5-9 are provided for auditing an old program library. Output is written to a temporary file and appended to the output (O) file at the end of the UPDATE. When the F parameter ('UPDATE,F.') is selected, options 5-9 apply to all decks on the old program library. If F is not selected, options 5-9 apply to decks listed on COMPILE directives only.

M=mrglfn merge input; allows two program libraries to be merged and written onto the new program library (see N option). If M=mrglfn is specified, the second old program library is on file mrglfn. If M is specified alone, the second old program library is on file MERGE. If omitted, there is no merge file.

N=npllfn new program library. If N=npllfn is specified, the new program library is written on file npllfn. If N is specified alone, a new program library is written on file NEWPL. If N is omitted, UPDATE does not generate a new program library.

O=outlfn list output file. List output is written on file lfn. If O is specified alone or omitted, list output is written on the file OUTPUT and is automatically printed.

P=opllfn old program library; ignored on creation run. If P=opllfn is specified, the old program library is on file opllfn. The default file is OLDPL.

Q quick update; takes precedence over F. Only decks specified on COMPILE directives are processed. Corrections other than ADDFILE that reference cards in decks not specified on COMPILE cards are not processed and UPDATE abnormally terminates after printing the unprocessed corrections. Q and F omitted is the normal (selective) UPDATE mode.

R=opt rewind files. Each option indicates a file to be rewound before and after the UPDATE run. The options are as follows:

C	Compile
N	New program library
P	Old program library and merge library
S	Source and PULLMOD.

If R is specified alone, no rewinds are issued for the program libraries, compile file, or source file. If R is omitted, the old and new program libraries, the compile file, and the source file are rewound before and after the UPDATE run.

S=srclfn source output. If S=srclfn is specified, the source output is written on file srclfn. If S is specified alone, the source output is written on file SOURCE. If omitted, UPDATE does not generate a source output file unless the source output is specified by T.

T=sxclfn source output exluding common decks; takes precedence over S parameter. If T=sxclfn is specified, source output excluding common decks is on file sxclfn. T alone specifies that source output excluding common decks is on file SOURCE. If T is omitted, no source output is generated unless source output is specified by S parameter.

U debug parameter. UPDATE execution is not terminated by normally fatal errors. If omitted, UPDATE execution terminates upon encountering a fatal error; this is the default.

W sequential new program library. The new program library (see N parameter) will be a sequential file. If omitted, the new program library will be determined by characteristics of the file specified by the N parameter.

X Compile file is in compressed format. If omitted, the compile file is not in compressed format. See the *UPDATE Reference Manual.*

Z compressed input file. The input file (see I parameter) is assumed to be in PCS compressed format. This parameter applies to the directives input file only; it does not apply to files specified by READ directives. If omitted, the input file is a normal, coded file.

8 The compile file output is composed of 80 column card images. If omitted, the compile file output is 90 column card images.

*=char The master control character (first character of each directive) for this UPDATE run is char; char can be A through Z, 0 through 9 or +, -, *, /, =, or $$$$. If omitted, the master control character is *.

/=char The comment control character for this UPDATE run is char; char can be A through Z, 0 through 9 or +, -, *, /, =, or $$$$. Note, however, that the character should not be changed to one of the abbreviated forms of directives unless NOABBREV is in effect. If omitted, the comment control character is /.

Examples: For examples and further documentation see the *UPDATE Reference Manual.*

7.16 Compilation and Assembly

This section discusses the control statements necessary for program compilation. The discussion is limited to descriptions of options available for each compiler; more complete information can be found in references given with each description.

7.16.1 APL

APL (A Programming Language) is an interpretive, mathematical processor that excels at the concise statement of mathematical operations. It has a large number of primitive operators, many of which would require several loops if written in another language. APL is primarily an interactive language.

The processor for the APL language is called with the APL control statement:

```
APL.
```

There are no parameters.

7.16.2 BASIC

BASIC is a simple, numerically-oriented language, which has special features for matrix operations. The CDC BASIC compiler is equally useful in batch and interactive modes.

The syntax for the BASIC control statement is as follows:

```
BASIC[,optional parameters].
```

AS
AS=0 — specifies ASCII mode. If AS=0 is specified, all input and output is performed in Old Mistic (OM) (see the *Interactive System User's Guide*, Section 5.1.1). This is the default.

If AS appears alone, all input and output is performed in ASCII Fancy (AF) (see the *Interactive System User's Guide*, Section 5.1.3).

B=binlfn
B=0 — specifies the name of the binary object file. The binary of the compiled program is written on binlfn. If binlfn is not specified, the default is BIN. If B=0 is given, no binary object file is produced; the program compiles into central memory; this is the default. The production of a binary object file requires at least 4000 additional words of memory.

BL — burstable listing. BL causes page ejects to appear between compiler output and the first line of execution output. The default suppresses these page ejects.

DB=opt debug and trace feature. 'opt' may be one of the following:

B causes binary generation or program execution regardless of compilation errors. (See B and GO parameters.) A program containing compilation errors will execute normally until a statement that caused the compilation errors is encountered.

DL activates program tracing as controlled by REM TRACE debug statements.

TR traces all statements regardless of REM TRACE debug statements.

0 debug and trace feature is not activated. This is the default.

If DB appears alone, both the B and DL options will be in effect.

If DB=0 or DB is omitted, the debug and trace feature is not activated. This is the default.

E=errlfn compile-time error diagnostics. Compiler error diagnostics are written on the file errlfn; the default is ERRS. If E is omitted, compiler diagnostics are written on listlfn as specified by the L parameter, or its default, OUTPUT.

EL=errlev list fatal or warning error diagnostics; errlev can be one of the following:

F list fatal diagnistics on file errlfn, as specified in the E parameter.

W both fatal and warning diagnostics are written on file errlfn. This is the default.

GO
GO=0 controls execution. If GO=0 is given, no execution will occur. If GO appears alone, the compiled BASIC program will execute, provided there were no compilation errors. If the B parameter was specified, the relocatable binary will be loaded and executed. (See DB parameter.) If the B parameter was not specified, the compiled-into-memory code will be executed.

If GO is omitted, the compiled BASIC program will execute without loading provided it was compiled into memory and there were no compilation errors. If the B option was specified, the compiled program will not be executed.

I=inlfn compiler input file. The BASIC source program is on the file inlfn. If I appears alone, the file COMPILE is assumed. The default is INPUT.

J=inpulfn
J=0 default input file for compiled BASIC program. When an 'INPUT' statement is executed, file inpulfn is read. If J appears alone inpulfn is INPUT. If J=0 is given, there is no default runtime input file. In this case, use of the 'INPUT' statement will abort the executing BASIC program.

K=prinlfn default output file for compiled BASIC program. When a 'PRINT' statement is executed, prinlfn is written. If K appears alone or is omitted, prinlfn is OUTPUT.

L=listlfn
L=0 compiler listable output file. Listable compiler output is written on file listlfn. If listlfn is not specified, the file OUTPUT is assumed; this is the batch default. If L=0 is specified, no compiler listing is produced; this is the interactive default.

LO=listopt listing options. More than one listopt may be specified; multiple values must be separated by slashes (/). 'listopt' may be one or more of the following:

S a source listing is produced on the file iistlfn specified by the L parameter. This is the default.

O an object listing is produced on the file listlfn specified by the L parameter.

0 no source listing is produced.

If LO appears alone, LO=S is assumed.

PD=6
PD=8 print density for listlfn (L parameter) and prinlfn (K parameter). If PD=6 is specified, print density is set to 6 lines per inch. If PD=8 is specified, print density is 8 lines per inch; this is the default.

PS=n page size. Establishes the listlfn (L parameter) page size as n printable lines per page (where 4≤n≤32768). This parameter has no effect on execution output. Lines are not counted at execution time.

If PD specifies a non-default print density, then PS is calculated as follows: PS=PD * default PS / default PD.

For more information, see the CDC *BASIC Version 3 Reference Manual* (Publication No. 19980300).

7.16.3 COBOL

The COBOL control statement calls the COBOL compiler, specifies the files to be used for input and output, and indicates all compiler options. Full documentation for COBOL is found in the *COBOL Reference Manual.*

```
COBOL[,optional parameters].
```

Parameters may be specified in any order.

A leading blanks are treated as zeros in arithmetic statements and comparisons. The default is no A.

B=binlfn
B=0 a relocatable binary file is written on the local file named binlfn. If B is given alone, or if the parameter is omitted, binary output is written on the LGO file. If B=0 is given, binary output is suppressed.

BUF provides compatibility with version 3 COBOL methods of determining buffer size. In COBOL v4, buffer sizes are based on the record description; a minimum size of 514 words has been established. The BUF parameter selects the version 3 method which does not use record description or ALTERNATE AREAS to determine the minimum block size.

C specifies that a copy be made from source (the COPY mode used in previous versions of COBOL) rather than from the library, which is the default.

When C is used, FROM LIBRARY should be added to the COPY statement to make a copy from the library; and FROM SOURCE need not be specified in the COPY clausė to copy a source item in the Data Division. This ensures that COBOL v3 source copies are correct.

D inhibits ėxecution of a COBOL program when an E diagnostic is encountered. When D is used and an E diagnostic occurs, compilation continues and an error message that inhibits loading is written on the relocatable binary file. The compiler displays the following message in the dayfile:

FATAL COBOL ERRORS OR D OPTION IN EFFECT

DB all subscript values are checked to determine whether they are in the range of the OCCURS clause for the item. When any subscript is outside the range, the diagnostic NEGATIVE, ZERO, OR HUGE SUBSCRIPT is given, and the source line causing the error is indicated.-The run is terminated. No subscript checking is done if DB is not selected.

DB1 allows the generation of object code which calls the paragraph trace feature to trace the flow of the program. Although DB1 does not itself cause a trace to be created, the parameter must be selected if the trace feature is to be entered by the COBOL source program. See Section II-14 of the COBOL 4 Reference Manual for a detailed description of the trace feature.

E=prog output of a COBOL compilation can be added to the system library with LIBEDIT; the V parameter is automatically assumed.

The name of the main overlay of the absolute program is specified by prog; it calls the program into ėxecution from the library. It must be 5 characters or less, and it must not duplicate the name in any PROGRAM-ID clause used, any ENTRY name, or any implementor name specified in a SELECT or SPECIAL NAMES clause.

This parameter causes an overlay card of the form OVERLAY (COBCODE, 00,00) to be generated (unless the OB parameter is specified). A load, nogo sequence creates the proper absolute file on COBCODE which can be processed by LIBEDIT.

F all data name entries described as COMPUTATIONAL are interpreted as COMPUTATIONAL-1.

H if no OPEN statement is executed during processing of a SORT input or output procedure, this parameter should be used to increase sort efficiency. If H is not included, all program files are allocated buffer space before the sort starts, resulting in unnecessary space reservation and a decrease in efficiency. If H is specified and a file is opened during execution of an input or output procedure, a diagnostic is issued.

I=inlfn identifies the source input file name; magnetic tape files must be BCD and even parity. If no inlfn is specified or if I is omitted, the file INPUT is assumed.

K=sslfn
K=0 This must be included if the user wishes to employ the facilities of CDCS. The file containing the sub-schema is named sslfn. K=0 indicates no sub-schema file is used in this compile; the CDCS facilities are not utilized.

L[op]=listlfn
L[op]=0

List source input on file listlfn. If listlfn is not given, or if L is omitted, the file OUTPUT is assumed. If L=0 is specified, list output is suppressed, except for C and E diagnostics.

op is any combination of the following options to provide features in addition to the normal listing.

X	Extended diagnostics; includes T, U, E, C
R	Data name and procedure name cross reference list
C	Items copied from library
O	Object code in octal
M	Data map

```
Examples: LXC=OUTFILE
          LR
```

N

to ensure an ANSI standard source program. When a non-ANSI feature is detected by the compiler, an E type diagnostic is issued.

OB=ovllfn

separates overlay segments from main programs so that separately compiled programs can be loaded properly. Binary output from overlay segments is placed on file ovllfn.

P

non-ANSI reserved words are allowed in the source program, the N parameter option is automatically selected to diagnose non-ANSI features, and ANSI input-output is assumed.

S=srclfn

specifies the name of the local file containing the source library information. If srclfn is not specified, or if S is omitted, the file COLIB is assumed.

SUB

this parameter is required when compiling subprograms (unless the SUBM parameter is used). SUB enables the subprogram and main program to be loaded together properly.

SUBM

is used when compiling a COBOL subprogram if the main program is written in a language other than COBOL. It cannot be used when the main program is a COBOL program, and can be specified for only one COBOL subprogram; other COBOL subprograms, if any, are compiled using the SUB parameter. SUBM enables the subprogram and and main program to be loaded together properly.

T

must be specified for a tape sort. It causes four files to be requested for each sort call: CSORT1, CSORT2, CSORT3, and CSORT4. These files may be assigned to disk.

U

with this parameter, the user can select a collating sequence other than the standard CDC default collating sequence. The U option uses the ASCII collating sequence for SORT verb executions and for HIGH-VALUE figurative constants. LOW-VALUE is always spaces regardless of the selected collating sequence.

V

if the loaded program is to be saved using NOGO with the file name specified, the V parameter must be specified for all COBOL/SORT programs. (Use of the E parameter automatically implies the V parameter flag as does segmentation in the program.) Specifying this parameter causes the SORT code to be included in the program rather than being loaded dynamically. This action is necessary whenever the program is stored in absolute form. Refer to the *COBOL Reference Manual*, Appendix E, for a description of SORT considerations.

W
an independent segment (priority number 50-99) normally may overlay or be overlaid by an overlayable fixed segment or another independent segment. In COBOL 4, and for ANSI programs, an independent segment is made available in its initial state. To override this usage and provide independent segments in their last used state so that COBOL 3 programs can be run without change, this parameter should be included.

Z
this parameter must be included to create an environment for running the object program that is compatible with files created under COBOL 3 and SCOPE Indexed Sequential Version 1. This turns on the C and W parameters.

On BCD devices, record type is set to Z by the Z parameter. It does not turn on the BUF parameter to assign large buffers.

Examples: For examples and further documentation, see the *COBOL Reference Manual*.

7.16.4 COMPASS

The COMPASS control statement causes the COMPASS assembler to be loaded and executed. Full documentation may be found in the CDC *COMPASS Reference Manual* (Publication No. 60360900); the following is a description of the assembler options.

```
COMPASS[,optional parameters].
```

A
abort mode; abort job and skip to an 'EXIT,S.' statement if any assembly errors occurred. If A is omitted, COMPASS will not abort for assembly errors.

B=binlfn
B=0
binary output. If B=binlfn is given, the binary will be written on the named file. If B=0 is given, no binary output will be generated. The default is LGO.

D
debug mode. Binary is generated on the file indicated by the B parameter in spite of assembly errors and regardless of the abort mode (A parameter). D is ignored if B=0 is specified.

If D is omitted, assembly errors will inhibit binary output. When the A parameter is present, no binary output is written at all for a subprogram containing assembly errors. If the A parameter is omitted, the message "ERRORS IN ASSEMBLY" is written to the file indicated by the B parameter for each subprogram containing assembly errors; this causes a fatal error at load time.

F=val
FORTRAN mode; establishes the value of the special element *F. F=number gives *F a one-digit decimal number. The default is 0.

F=name can be one of the names listed below; *F will have the number corresponding to name.

name	*F value
COMPASS	0
FTN	2

G=syslfn
G=syslfn/ovl
G=0

get system text. G=syslfn means load the first system text overlay, if any, in the specified sequential binary file. The default file is SYSTEXT. If G=syslfn/ovl is specified, the sequential binary file is searched for a system text overlay whose name is ovl; the first such overlay found is loaded. If G=0 is given, no system text is loaded from a sequential binary file.

See also the discussion of multiple system text overlays below.

I=inlfn

source of assembler input. If I=inlfn is given, the source deck is on the named file. If I is given alone, the source deck is on file COMPILE in either compressed or expanded format. If I is omitted, the file INPUT is assumed.

L=listlfn
L=0

full list on file listlfn. If listlfn is not specified, the file OUTPUT is assumed. If L=0 is given, no full list will be generated.

When the full list is on a different file than the short list (O parameter), the listing for each subprogram is preceded by a one word header consisting of an asterisk and the first six characters of the subprogram name. This header identifies the subprogram as a convenience for sorting and cataloging. See also the O parameter below.

L6 or L8

Print listing at 6 or 8 lines per inch. The default is L8.

LO=opt
LO=$$$$
LO=0

list options; selects or deselects a maximum of nine of the following list options:

A	assembled statements
B	binary control statements
C	listing control statements
D	include details
E	include echoed lines
F	IF skipped lines
G	generated code, including source code
J	generated code (without source code)
L	master list control
M	macros and opdefs
N	nonreferenced symbols
R	reference map
S	systems macros/opdefs
T	nonreferenced system symbols
X	XTEXT lines

(For a description of each list option, see Section 4.11.1 of the CDC *COMPASS Reference Manual.*) If LO is omitted or LO=0 is specified, then B, L, N, and R are selected. If LO is specified alone, list options C, F, G and X are selected, and R is deselected.

The options list in LO=opt can be 1-9 characters concatenated. Inclusion of B, L, N, or R deselects the corresponding option. Otherwise, inclusion of a character selects the option.

IF LO=$$$$, all list options are selected. (The parameter is actually a single $; however, the system requires that this be represented on a control statement as shown.)

ML=modlev

initial value of MODLEVEL micro. If ML=modlev is given, MODLEVEL is defined as the string 'modlev' (9 characters maximum, special characters delimited by $) at the start of each assembly. IF ML is specified alone or omitted, MODLEVEL is defined equal to JDATE.

N

no eject; suppresses ejects caused by normal listing control. The only page ejects are at the beginning of new subprograms. IF N is omitted, normal ejects will occur.

NN

Each page eject is replaced by a double space and a line of asterisks in place of page boundaries.

O=outlfn
O=0

short list; suppressed if full list is directed to the same file or if no assembly errors occur. However, if the full list and short list are on different files (for example, the full list is written on OUTPUT and the short list is written on the named file), the short list will contain all error lines. If O=outlfn is specified, the list output will be written on the file outlfn. The default is OUTPUT. If O=0 is specified, no short list will be generated.

P

continue page numbering. Page numbering continues from subprogram to subprogram. If P is omitted, page numbering begins with 1 at the start of each subprogram.

PC=strng

initial value of PCOMMENT micro. If PC=strng is specified, PCOMMENT is defined as strng at the start of each assembly. Characters are truncated from the right or blanks are appended to the right, as necessary, so that the length of the micro value is exactly 30 characters. If PC is specified alone or omitted, PCOMMENT is defined as 30 blanks at the start of each assembly.

S=ovl
S=lib/ovl
S=0

system text name. ovl is a system text overlay. lib is a library name; lib may be a user library or system library. If S is omitted, and if there are no G parameters other than G=0, the overlay named SYSTEXT will be loaded from the job's current global library set. SYSTEXT will be loaded if S is specified alone. If S=ovl is specified, the system text overlay named ovl is loaded from the job's current global library set. If S=lib/ovl is specified, the system text overlay ovl is loaded from the library lib.

See also the discussion of multiple system text overlays below.

X=extlfn

source of external text (XTEXT) when location field of XTEXT pseudo instruction is blank. If X=extlfn is specified, the external text is on the named file. If X is specified alone, the external text is on file OPL. If the X parameter is omitted, the external text is on file OLDPL.

Y=WARN

flag MSU-only features as Y-errors. This includes the IOR and AND pseudo-ops. Y-errors are fatal if Y is used alone; they are non-fatal if Y=WARN is used.

Multiple System Text Overlays

COMPASS 3 allows up to seven system text overlays to be used for an assembler run. They are specified by G and S parameters on the COMPASS control statement. Each G parameter (except G=0) specifies loading of a system text overlay from a sequential binary file, and each S parameter (except S=0) specifies loading of a system text overlay from a user library file or a system library. The G and S parameters can be used in any combination and in any order, and can be intermixed freely with other parameters, provided the total number of system text overlays specified does not exceed seven. COMPASS loads the system text overlays in the order in which the G and S parameters occur on the COMPASS statement. If a system macro, micro, or symbol is defined by more than one system text, only the last definition is used. See the CDC *COMPASS Reference Manual.*

Examples:

```
COMPASS,I,S,S=PFMTEXT,G=MYTEXT.
```

Reads source from file COMPILE and gets system text from overlays SYSTEXT and PFMTEXT in the global library set, and from the local file MYTEXT.

```
COMPASS,G=FILE/SCPTEXT,S=MYLIB/TEXT.
```

Get system from overlay SCPTEXT on the file FILE, and from overlay TEXT in library MYLIB.

For more examples, see the *COMPASS Reference Manual.*

7.16.5 FTN 4

The FORTRAN Extended version 4 compiler is executed by the FTN control statement. The FTN control statement calls the compiler, specifies the files to be used for input and output, and indicates the types of output to be produced.

Full documentation of FTN 4 is found in the CDC *FORTRAN Extended Version 4 Reference Manual* (Publication No. 60497800). The control statement options are described below.

```
FTN[,optional parameters].
```

Unrecognized parameters are ignored. Conflicting options either are resolved or cause compilation to terminate, depending on the severity of the conflict; this resolution is indicated in a dayfile entry.

The values of the A, B, D, G, I, L, ML, P, PD, S, and X parameters are passed to COMPASS when intermixed COMPASS subprograms are present.

A
A=0 — exit parameter. If fatal errors have occurred during compilation, the system aborts the job, and control passes to the next 'EXIT,S.' control statement. If no such control statement is found, the job is terminated. A takes precedence over GO but not over D.

If A=0 is specified, the system advances to the next control statement at the end of compilation whether or not fatal errors have been found. This is the default.

B=binlfn
B=0

binary object file. If B=binlfn is specified, generated binary object code is written on file binlfn. LGO is the default file. If B=0 is given, no binary object file is produced. B=0 cannot be specified with GO.

The B parameter conflicts with the Q and E parameters.

BL
BL=0

burstable listing. BL generates output listing that is easily separable into components by issuing page ejects between source code, error summary (if present), cross reference map, and object code (if requested); it ensures that each program unit listing contains an even number of pages by issuing a blank page at the end if necessary.

C
C=0

COMPASS assembly. C selects the COMPASS assembler to process the symbol object code generated by FTN. When the C parameter is specified, FTNMAC is selected for the system text for the COMPASS assembly. Therefore, if the C parameter is selected, the maximum number of system texts that can be specified with the G and S parameters is six.

C=0 selects the FTN internal assembler which is two to three times faster than the COMPASS assembler. This is the default.

The C parameter conflicts with the TS, Q, and E parameters.

D=dbglfn
D=0

debugging mode parameter. D=dbglfn must be specified if the debug utility described in the *FORTRAN Reference Manual* is to be used. The file dbglfn is where the user debug deck resides. D alone implies D=INPUT: D alone reads from INPUT regardless of the file specified by the I parameter. Binary object code is generated on the file indicated by the B parameter regardless of compilation errors or the exit parameter A. Interspersed COMPASS code, if present, is assembled under the COMPASS D option. Specifying D automatically activates OPT=0 and the T option; thus, 'FTN,D.' is equivalent to 'FTN,D,OPT=0,T,A=0.' If D=0 is specified, debug statements are ignored. This is the default.

OPT=1 or OPT=2 is ignored if D or D=lfn is specified. The D parameter conflicts with the TS parameter.

E=editlfn
E=0

editing parameter. If E=editlfn is given, generated object code is output as COMPASS line images on the file editlfn, which is rewound at the end of compilation. E alone implies E=COMPS. Each program unit is prefaced with the line image, '*DECK program', so that the file will be suitably formatted for input to UPDATE. Binary object code is not produced; and COMPASS is not called. When the file editlfn is assembled subsequently, S=FTNMAC must be specified on the COMPASS control statement.

If E=0 is specified, the object file is generated in normal binary code rather than as COMPASS line images. This is the default.

The E parameter conflicts with the B, C, GO, OL, TS, and Q options.

EL=errlev

listing error diagnostics. The errlev can be one of the following:

A List diagnostics indicating all non-ANSI usages, as well as fatal diagnostics. Also, list informative diagnostics if compiling under OPT=0, 1, or 2; list note and warning diagnostics if compiling in TS mode.

I List informative and fatal diagnostics if compiling OPT=0, 1, or 2; list note, warning, and fatal diagnostics if compiling in TS mode. This is the default.

N List note, warning, and fatal diagnostics if compiling in TS mode; list fatal diagnostics if compiling under OPT=0, 1, or 2.

W List warning and fatal diagnostics if compiling in TS mode; list fatal diagnostics if compiling under OPT=0, 1, or 2.

F List fatal diagnostics.

ER
ER=0

error recovery. IF ER is specified, code is generated for object-time reprieve. When this option is selected, any of the following execution time errors are reprieved: arithmetic mode error, bad system request in RA+1, CP or I/O time limit exceeded, mass storage limit exceeded, or an operator drop. The name of the program unit in which the error occurred is written to the job dayfile along with the line number where the error occurred. This option increases the size of object code and should be used only while a program is being debugged. This is the default if in TS mode or if OPT=0.

If ER=0 is specified, no code is generated for object-time reprieve. This is the default if OPT=1 or OPT=2.

G=syslfn
G=syslfn/ovl
G=0

get system text file. G=syslfn loads the first system text overlay from the sequential binary file lfn. G alone implies G=SYSTEXT. G=syslfn/ovl searches the sequential binary file syslfn for a system text overlay named ovl and loads the first such overlay encountered. G=0 prevents system text loading from the sequential binary file. This is the default. A maximum of seven system texts can be specified by any combination of the G, S, and C parameters.

Note: This feature is for COMPASS subprograms only.

GO
GO=0

automatic execution (load and go). If GO is specified, the binary object file is loaded and executed at the end of compilation. If GO=0, the binary object file is not loaded and executed. This is the default.

The GO parameter conflicts with the Q, E, and B=0 options.

I=inlfn

source input file. Source code to be compiled appears on file inlfn. File INPUT is the default. Compilation ends when an end-of-partition, end-of-information, or end of SCOPE section is encountered. I alone implies I=COMPILE.

L=listlfn
L=0

list source input and listable output (specified by list control options BL, EL, OL, R, and SL) on file listlfn. If list control options are not specified, the listing consists of the source program, and informative and fatal diagnostics. If listlfn is not given, the file OUTPUT is assumed.

If L=0 is specified, fatal diagnostics and the statement that caused them are listed on the file OUTPUT. All other compile time output, including intermixed COMPASS, is suppressed. List control options are ignored.

ML=nnn

modlevel. ML=nnn specifies nnn as the value of the MODLEVEL micro used by COMPASS. The value nnn consists of 1 to 7 letters or digits. If ML is specified alone, the current date in the form 'yyddd' (where yy is the year and·ddd is the number of day within the year) is used for the MODLEVEL micro. This is the default.

OL
OL=0

object list. Generated object code is listed on the list output file. If OL=0 is specified, the object code is not listed. This is the default.

The OL option conflicts with the Q and E options.

OPT=n

optimization parameter; n may be one of the following:

0 fast compilation (automatically activates T and ER options)
1 standard compilation and execution. This is the default.
2 fast execution

If n is not specified, OPT=2 is assumed.

P
P=0

pagination parameter. If P is specified, page numbering of output listing is continuous from subprogram to subprogram, including intermixed COMPASS output. If P=0 is specified, page numbering begins at 1 for each subprogram. This is the default.

PD=6
PD=8

print density of 6 or 8 lines per inch. If PD=6, compile time lists are produced at a density of six lines per inch. If PD=8, compile time listings are produced at a density of eight lines per inch if printed on a central site printer (source "B"). This is the default. PD alone implies PD=8.

PL=n

print limit. The value n is the maximum number of lines that can be written at execution time on the file OUTPUT; n must not exceed ten characters. If n is suffixed with the letter B, it is assumed to be octal and must not exceed 777 777 777B; otherwise, it is assumed to be decimal and must not exceed 9 999 999 999. The default is PL=50000.

The PL parameter is operative only when appearing on an FTN statement to compile a main program.

The print limit (specified at compilation time either explicitly or by default) can be overridden at execution time by a parameter of the same format appearing on the LGO or EXECUTE control statement.

PMD
PMD=0

post mortem dump. It provides interpreted output in a form which is easier to understand than the octal dump normally output following a fatal error. PMD=0 is the default (i.e. no PMD generated). Use of Post Mortem Dump does not affect the use of FORTRAN Extended DEBUG or Cyber Interactive Debug. Post Mortem Dump is activated only after a fatal error has occurred.

PS=n

page size. The value n is the maximum number of lines per page for compiler listings (including headers). If n < 4, the default value is substituted. The default is PS=60 if PD=6; PS=80 if PD=8.

PW=nn

page width. nn is the number of characters on a line of listable output. Values less than 50 or greater than 136 are diagnosed and ignored. The default for a printer output file is 126. PW alone implies PW=72. This is the default if on a terminal output file. The PW option is valid only with TS mode.

Q
Q=0

program verification. Q given alone specifies quick mode: the compiler performs full syntactic scan of the program, but no object code is produced. No code addresses are provided if a reference map is requested. This mode is substantially faster than a normal compilation; but it must not be selected if the program is to be executed.

Q=0 gives normal compilation. This is the default.

The Q option conflicts with the B, C, GO, OL, TS, and E options.

R=n

symbolic reference map listed. The value n may be one of the following:

0 no map produced. This is the default.
1 short map (symbols, addresses, properties, DO loop map).
2 long map (short map plus references by line number).
3 long map plus listing of common block members and equivalence classes.

R alone implies R=2.

In TS mode, R=3 is identical to R=2; common and equivalence classes are not listed.

ROUND=op
ROUND=0

rounded arithmetic computations. If ROUND=op is given, op is any combination of the arithmetic operators + - * /. Single precision real and complex floating point arithmetic operations are performed using the hardware rounding feature, as described in the CDC *6000 Computer Systems Reference Manual*. ROUND alone implies ROUND=+-*/. If ROUND=0, computation for the indicated operators is not rounded. This is the default.

The ROUND option controls only the in-line object code compiled for arithmetic expressions; it does not affect computations of library subprograms or input/output routines.

S=ovl
S=lib/ovl
S=0

system text (library) file. ovl is a system text overlay. lib is the name of a user library file or system library. If S=ovl is given, ovl is loaded from the job's current library set. If S=lib/ovl is given, ovl is loaded from lib.

If S=0 is specified, system text file is not loaded when COMPASS is called to assemble any intermixed COMPASS programs. This is the default if the G parameter is other than G=0. S alone implies S=SYSTEXT. This is the default if G=0 is specified.

The S parameter is for COMPASS subprograms only. Up to 7 system texts can be specified by repeating this option.

SEQ
SEQ=0

sequenced input. IF SEQ is specified, the source input file is in sequenced line format (see the *FORTRAN Extended Reference Manual*, Section III-15).

If SEQ=0 is specified, the source input file is in standard FORTRAN format. This is the default.

Specifying the SEQ parameter automatically activates the TS option; sequenced line format is not recognized by the optimizing compiler or COMPASS. The SEQ parameter conflicts with the OPT=0, 1, or 2 options.

SL
SL=0

source list. If SL is given, the source program is listed on the file specified by the L parameter. This is the default. If SL=0, the source program is not listed.

STATIC
STATIC=0

If STATIC is given alone, dynamic memory management is inhibited at execution time by CRM. The computer generates a set of 'LDSET,USE' directives specifying each of the capsules needed by the program. The specified library programs are then statically loaded. STATIC is required for any program that dynamically extends blank common. STATIC should not be used if MSU Record Manager (MSURM) is used. MSURM will use STATIC mode regardless of the FTN compiler control statement setting.

If STATIC=0 is specified then no special LDSET directives are generated and CRM uses dynamic memory management at execution time. This option results in a decrease in field length needed at execution time. This is the default.

SYSEDIT
SYSEDIT=0

system editing. IF SYSEDIT is given alone, all input/output references are accomplished indirectly through a table search at object time. File names are not entry points in the main program, and subprograms do not produce external references to the file name.

If SYSEDIT=0 is specified, input/output references are accomplished directly; file names are used as entry points in the main program, and subprograms produce external references to the file name. This is the default.

The SYSEDIT feature is used primarily for relocatable programs residing on loader libraries.

T
T=0

error traceback. If T is specified, full error traceback occurs when an error is detected. Calls to basic external functions are made with call-by-name sequence.

T=0 means no traceback occurs when an error is detected. Calls to basic external functions are made with the more efficient call-by-value sequence. A saving in memory space and execution time is realized. This is the default.

The T option is provided to assist in debugging programs. Selecting the D parameter or OPT=0 automatically activates the T option.

TS

timesharing mode. In TS mode, compilation speed and field length are optimized at the expense of execution speed and field length. TS mode is preferable to the optimizing compilation modes (OPT=0, 1, or 2) for the debugging stages of a program. Specifying option TS together with option C, D, E, or Q constitutes a fatal control statement error. If TS is specified, any OPT parameters are ignored. IF TS is omitted, OPT=1 is assumed.

UO
UO=0

unsafe optimization parameter. UO allows the compiler to perform certain optimizations which are potentially unsafe. UO is ignored unless OPT=2 is also specified.

UO=0 means unsafe optimization is not performed. This is the default.

X=extlfn

external text name. File extlfn is the source of external text (XTEXT) when the location field of the XTEXT pseudo instruction is blank. Only one X parameter may be specified. The default is OLDPL. X alone implies X=OPL. The X feature is for COMPASS subprograms only.

Z=0 zero parameter. All subroutine calls having no parameters are forced to pass a parameter list consisting of a zero word. This feature is useful to COMPASS-coded subroutines expecting a variable number of parameters. Z should not be specified unless necessary, since programs require less memory if Z is omitted. If Z=0 is given, the zero parameter is not passed. This is the default.

Example:

FTN.

Selects the following options:

A=0
B=LGO
BL=0
C=0
D=0
E=0
EL=I
ER=0
G=0
GO=0
I=INPUT

L=OUTPUT
MS=yyddd
OL=0
OPT=1
P=0
PD=8
PL=50000
PMD=0
PS=80
PW=126
Q=0

R=0
ROUND=0
S=SYSTEXT
SEQ=0
SL
STATIC=0
SYSEDIT=0
T=0
TS=0
UO=0
X=OLDPL
Z=0

7.16.6 FTN 5

The FORTRAN version 5 compiler is called and executed by the FTN5 control statement. The FTN5 control statement calls the compiler, specifies the files to be used for input and output, and indicates the type of output to be produced. The control has the following form:

FTN5[,optional parameters].

ANSI
ANSI=0
ANSI=opt

ANSI diagnostics. Specifies whether use of non-ANSI extensions to FORTRAN are to be diagnosed and if so, how severely. opt may be one of the following:

F Any non-ANSI use will result in a fatal error. All ANSI warning diagnostics become fatal.

T ANSI errors are treated as trivial errors.

When ANSI is specified alone, ANSI=T is assumed.

If ANSI is omitted, ANSI=0 is assumed; no ANSI diagnostics are generated.

ARG
ARG=0
ARG=opt[/opt]

Argument list attributes. Specifies attributes of external procedure argument lists generated by the compiler. Opt may be one of the following:

COMMON Argument lists generated for external procedures will be in the proper form for interlanguage communication. Specification of COMMON implies -FIXED; that is, FIXED is not selected.

FIXED	All references in the FORTRAN program to a given external procedure have the same number of arguments. (The compiler-generated argument lists will not contain a zero terminator.)

Note: Minus sign (-) is a legal character.

If ARG is specified alone, ARG=-COMMON/FIXED; that, COMMON is not selected.

If ARG is omitted, ARG=0 is assumed; neither option is selected. This is the default.

The initial value is ARG=0. Specification of ARG=COMMON/FIXED is not permitted. If ARG=FIXED is not selected, a zero word terminates the list.

B
B=0
B=binlfn

Binary output file. Specifies the name of the compiler output file.

If B=0 is specified, no binary file is produced.

When B=binlfn is specified, computer-generated binary code is output on the file binlfn.

If B is specified alone, B=BIN is assumed. However, if B is omitted, B=LGO is assumed. This is the default.

BL
BL=0

Burstable listing. Controls page ejects in the listing produced by the compiler.

If BL=0 is specified, the page ejects are minimized, and listings are generated in compact format.

BL, specified alone, generates output listings which are easily separable into components by issuing page ejects between source listing, cross-reference-attributes map, and object code listing.

If BL is omitted, BL=0 is assumed. This is the default.

CS
CS=opt

Collating sequence. Specifies the weight table to be used for the evaluation of character relational expressions. opt may be one of the following:

FIXED	A fixed weight table is required. (See Collating Sequence Control)
USER	A user-specified weight table is permitted.

If CS is specified alone, CS=FIXED is assumed. If CS is omitted, CS=USER is assumed. This is the default.

DB
DB=0
DB=opt

Debugging options. The DB parameter is a multiple binary value parameter that selects debugging options. opt may be any of the following:

ER — Selects object time reprieve of execution errors. A message identifying the program unit and line number containing the error is printed.

ID — A line number table, symbol table, and special stylized object code are generated along with the binary code. DS=-ID overrides any previous DEBUG control statement specification. DB=ID requires OPT=0.

PMD — Generates symbol tables needed by POST MORTEM dump so that a symbolic analysis of error conditions, variable names and values, and traceback information can be written to an output file.

SB — Subscript bounds checking is performed.

SL — Character substring expressions are checked to ensure that the substring references are within the string.

ST — Same as DB=ID except that the stylized object code is not generated.

TB — A full error traceback occurs upon detection of an execution time error. This option causes arguments to intrinsic functions to be passed by reference.

If DB=0 is specified, no options are selected. If DB is omitted, DB=0 is assumed. This is the default.

If DB is specified alone, DB=TB/SB/SL/ER/PMD is assumed. If DB=PMD is specified, ARG=FIXED must not be selected.

DO
DO=0
DO=opt

DO loop control. Specifies the manner in which DO loops are to be interpreted by the compiler. opt may be one of the following:

LONG — Permits the trip count to exceed 131071.

OT — Sets the minimum trip count for DO loops to one. This option can result in faster program execution, but should only be used if no trip count=0 is specified.

If DO is specified alone, DO=OT is assumed. If DO=0 is specified, the trip count must be less than 131071 and minimum trip defaults to zero.

If DO is omitted, DO=0 is assumed. This is the default.

DS
DS=0

Directive suppression. Suppresses the recognition of C$ directives.

If DS is specified alone, all C$ directives are treated as comments.

If DS=0, all C$ directives are recognized and processed. If DS is omitted, DS=0. This is the default.

E
E=errlfn

Error file. Specifies the name of the file to receive error information.

The errlfn option specifies that in the event of an error of EL-specified severity or higher, the error line and diagnostic are written to errlfn. If the L parameter is specified, this information is also written to the file specified by the L parameter.

The E parameter, when specified alone, is equivalent to E=ERRS. When E is omitted, E=OUTPUT is assumed; this is the default.

EL
EL=opt

Error level. Indicates the severity level of errors to be printed on the output listing. The levels are ordered by increasing severity. Specification of a particular level selects that level and all higher levels. opt may be one of the following:

T — Lists trivial errors. The syntax of these errors is correct but the usage is questionable.

W — Lists warning errors. These are errors where the syntax is incorrect but the compiler has made an assumption (such as inserting a comma) and continued.

F — Lists fatal errors. A fatal error prevents the compiler from processing the statement where the error occurred.

C — Lists catastrophic errors. These errors are fatal to compilation; the compiler is unable to continue processing the current program unit. Compilation continues with a subsequent program unit.

If EL is specified alone, EL=F is assumed. If EL is omitted, EL=T is assumed; this is the default.

ET
ET=0
ET=opt

Error terminate. Specifies the action to be taken by the compiler when compilation has completed. If an error of the specified level or higher occurs, the job "EXIT,S" control statement. opt may be one of the following:

T — Skips if errors of severity T or higher are detected.

W — Skips if errors of severity W or higher are detected.

F — Skips if errors of severity F or higher are detected.

C — Skips if errors of severity C are detected.

If ET=0 is specified, the job continues even if errors are encountered.

If ET is specified alone, ET=F is assumed. If ET is omitted, ET=0 is assumed; this is the default.

G
G=0
G=opt

Get system text file. Specifies the name of a file to read to obtain a system text for intermixed COMPASS subprograms. opt may be one of the following:

syslfn — Loads the system text from file syslfn.

syslfn-secname — Loads the system text from section secname on file syslfn. (The hyphen is required separator notation.)

If G=0 is specified, then no system text is loaded. If G is specified alone, G=SYSTEXT is assumed.

If G is omitted, G=0. This is the default.

GO
GO=0

Automatic execution. Specifies automatic loading and executing. The GO parameter, when specified alone specifies that the binary output file is loaded and executed after compilation.

If GO=0, the binary output file is not loaded and and executed after compilation.

If GO is omitted, GO=0 is assumed. This is the default.

I
I=inlfn

Input file. Specifies the name of the file containing the input source code.

If I=inlfn is specified, source code to be compiled is contained in file inlfn. Compilation ends when an end-of-section, end-of-partition, or end-of-information is encountered. I=0 is not allowed.

The I parameter, when specified alone, is equivalent to I=COMPILE. If I is omitted, I=INPUT is assumed. This is the default.

L
L=listlfn

List file. Specifies the name of the file where the compiler writes the source listing and any other requested listing information except diagnostics. L=0 suppresses all listings except that directed to the E file.

If L=listlfn is specified, listing is on listlfn.

If is specified alone, L=LIST is assumed. If L is omitted, it is equivalent to L=OUTPUT; this is the default.

LO
LO=0
LO=opt[/opt]

Listing options. Specifies the information that is to appear on the output listing file. Multiple options can be specified. opt may be one of the following:

A — A list of program entities (variables, common blocks) and their attributes (data type, class, etc.) is written to the output file.

M — A map, showing the correlation of program entities and physical storage, is written on the output file.

O — Output object code (COMPASS mnemonics) is listed on the output file.

R — A cross-reference map is written to the output file.

S — A source listing of the program is written to the output file.

If LO=0 is specified, no O, R, A, M, or S information appears on the output listing.

If LO is specified alone, LO=S/A/R. However, if the LO parameter is omitted, LO=S/A. This is the default.

ML
ML=0
ML=str

MODLEVEL micro. Specifies the value of the MODLEVEL micro used by COMPASS.

If ML=str is specified, the string str is used for MODLEVEL micro; str consists of 1 through 7 letters or digits.

When ML=0 is specified, the current date, in the form yyddd (where yy is the year and ddd is the number of the day within the year), is used for the MODLEVEL micro.

If ML is specified alone or omitted, ML=0. This is the default.

OPT
OPT=n

Optimization level. Specifies the level of optimization performed by the compiler. n may be any of the following:

0 — Minimum optimization is performed, resulting in fastest compilation. OPT=0 is required for DB=ID. This is the default.

1 — Intermediate optimization is performed.

2 — High optimization is performed, resulting in slower compilation.

3 — Potentially unsafe optimizations in addition to all OPT=2 optimizations are performed.

If OPT is specified alone, OPT=2. If OPT is omitted, OPT=0; this is the default.

PD
PD=n

Print density. Specifies print density for all printable output (L and E files). The destination printer must be capable of supporting the specified density. For interactive connected files, PD options are suppressed. n may be one of the following:

6 Compiler output is printed at six lines per inch, single spaced.

8 Compiler output is printed at eight lines per inch, single spaced.

If PD is specified alone, PD=8 is assumed. However, if PD is omitted, PD=8 is assumed. This is the default.

PL
PL=n

Print limit. Specifies the maximum number of records (print lines) that the executing program can write to file OUTPUT. This parameter is operative only when appearing on an FTN5 statement used to compile a main program.

If PL=n is specified, output must not exceed n lines, n is a decimal integer consisting of one through 10 digits.

When the PL parameter is specified alone, PL=50000. If PL is omitted, PL=5000. This is the default.

PN
PN=0

Pagination. Specifies page numbering options for the compiler output listing.

If PN=0 is specified, page numbers begin at 1 for each program unit.

If PN is specified alone, page numbering is continuous from program unit to program unit, including intermixed COMPASS output.

If PN is omitted, PN=0 is assumed.

PS=n

Page size. The PS=n parameter specifies the maximum number of printed lines to be included on a page of output; n must not be less than 4.

If PS is omitted, PS=60 if PD=6; PS=80 if PD=8. These are the defaults.

PW
PW=n

Page width. Specifies the width of an output line.

If PW=n is specified, printed lines are to contain n characters; n is a deimal integer and must not be less than 50 or greater than 136. Lines shorter than 136 characters are reformatted rather than truncated.

If the PW parameter is specified alone, PW=72. The lines are folded for connected files in this case, not truncated.

If PW is omitted, PW=72 is the default for a connected listing or error file. For all other output files, PW=136 is the default.

QC
QC=0

Quick syntax check. Specifies that the computer is to perform a quick syntax check of the source program. When the QC parameter is specified alone, the compiler performs a full syntactic scan of the program, but no binary code is produced. No code addresses are provided if a reference map is requested. QC compilation is substantially faster than normal; but it must not be selected if the program is to be executed.

If QC=0 is specified, a quick syntax check is not performed. If QC is omitted, QC=0 is assumed. This is the default.

REW
REW=0
REW=opt[/opt]

Rewind files. Specifies the files to be rewound prior to compilation. opt may be any of the following:

B	Rewinds the binary output file (specified by the B parameter).
E	Rewinds the error file (specified by the E parameter).
I	Rewinds the input file (specified by the I parameter).
L	Rewinds the output file (specified by the L parameter).

If REW=0 is specified, no files are rewound. But if REW is specified alone, REW=I/B.

If the REW parameter is omitted, REW=0. This is the default.

ROUND
ROUND=0
ROUND=opt[/opt]

Rounded arithmetic options. Specifies which arithmetic operations are tobe performed using rounded arithmetic. This parameter controls only the inline object code compiled for arithmetic expressions; it does not affect computations performed by library subroutines, intrinsic functions, or input/output routines. opt may be one of the following:

A	All addition operations are rounded.
D	All division operations are rounded.
M	All multiplication operations are rounded.
S	All subtraction operations are rounded.

If ROUND=0 is specified, no rounding is performed.

ROUND specified alone implies ROUND=A/S/M/D. If ROUND is omitted, ROUND=A/S/M is assumed; this is the default.

S
S=0
S=x

System text file. Specifies the name of the system text to be read by the compiler.

x may be one of the following:

sname	Specifies the system text name to be sname and searches the global library set. The default sname is SYSTEXT.
lib-sname	Searches the library named lib for the system text named sname. (The hyphen separating lib and sname is required.)

If S=0 is specified, the system text file is not loaded when COMPASS is called to assemble any intermixed COMPASS subprograms.

If S is specified alone or omitted, S=SYSTEXT is assumed if G is not specified. However, if S is omitted and G is specified, S=0 is assumed.

SEQ
SEQ=n

Sequenced input. If the SEQ parameter is specified alone the source input file must be in sequenced line format.

If SEQ=0 is specified, the source input file is in standard FORTRAN format.

If SEQ is omitted, SEQ=0 is assumed. This is the default.

X
X=extlfn

External text name. Specifies the name of the file from which the COMPASS assembler reads the external text when it encounters an XTEXT directive in the intermixed COMPASS program.

If x=extlfn is specified, the COMPASS assembler reads external text from file lfn.

If X is specified alone, X=OPL. Howver, if X is omitted, X=OLDPL. This is the default.

Example:

```
FTN5.
```

Selects the following options:

ANSI=0
ARG=COMMON
B=LGO
BL=0
CS=USER
DB=0
DO=0
DS=0
E=OUTPUT
EL=T

ET=0
G=0
GO=0
I=INPUT
L=OUTPUT
LCM=D
LO=S/A
ML=0
OPT=0
PD=6
PL=5000

PN=0
PS=60 (if PD=6)
PS=80 (if PD=8)
PW=136 (PW=72 for connected file)
QC=0
REW=0
ROUND=A/S/M
S=SYSTEXT, if G is omitted
S=0, if G is specified
SEQ=0
X=OLDPL

For a complete description of FTN version 5, see the *FORTRAN version 5 Reference Manual* (CDC 60481300).

7.16.7 F45 Conversion Aid

The F45 conversion aid program converts valid FORTRAN 4 source images on a specified file to FORTRAN 5 statements. This program should only be used on programs which have compiled and run successfully under FORTRAN 4.

The conversion program flags any statements it cannot convert.

Output from the conversion program includes a listing and a source output file. The complete listing contains the original programs, the converted programs, the Update/Modify directives, and messages.

The calling sequence for F45 is:

```
F45[,optional parameters].
```

where the parameters are:

CC
CC=C
CC=*

comment control. If CC=C is specified, or is omitted, change $ indicating a comment line to C.

If CC=* is specified or if CC is specified alone, change $ indicating a comment line to +.

CI
CI=0
CI=idname

correction identifier for UPDATE/MODIFY (used with a COMPILE file as input). Unless at least one of the options LO=M, LO=F, PO=M, or PO=F is specified, the CI parameter is ignored.

If CI=0 is specified, an *IDENT directive is not generated, even if LO=M, LO=F, PO=M, or PO=F is specified.

If CI=idname is specified, an UPDATE/MODIFY directive is is generated. It will have the following format:

```
*IDENT idname
```

The idname is the correction identifier to be assigned to this set.

If CI is omitted or if CI is specified alone, an UPDATE/MODIFY directive is generated. The directive has the following format:

```
*IDENT dddhhmm
```

where ddd is the number of the day of the year. The hh is the hour of the day and mm is the minutes.

DD
DD=0

delete C$ directives. If DD=0 is specified, statements with a C$ in columns 1 and 2 are converted to comments by replacing the $ with a blank.

If DD is specified alone or if DD is omitted, all statements with a C$ in columns 1 and 2 are deleted.

ET
ET=0

exit termination. If ET=0 is specified or if ET is omitted, termination is normal, no matter what conditions exist in the input file.

If ET is specified alone, termination is normal only if none of the following conditions exist:

1. FORTRAN syntax errors,
2. statements requiring manual action (not just inspection),
3. requests for UPDATE/MODIFY output files when input is not a COMPILE file.

The Conversion Aid will abort if any of these conditions exist, and then system error exit processing takes over.

I
I=inlfn

source input file. If I=inlfn is specified, source programs are read from the file named inlfn.

If I is specified alone, it is assumed that I=COMPILE.

If I is omitted, it is assumed that I=INPUT.

L
L=0
L=listlfn

listing file. If L=listlfn is specified, listings are written on the file named listlfn.

Ir L=0 is specified, no listing is produced.

If L is specified alone, listings are written on the file named LIST.

LO
LO=opt

listing options. opt may be one of the following:

E Produces an error listing containing lines requiring manual action and related Conversion Aid messages and error diagnostics.

F Output is similar to the LO parameter.

M Produces a modification listing containing generated UPDATE/MODIFY directives, translated and added lines, lines requiring manual action, and messages and error diagnostics.

S Output is similar to the default parameter.

If LO is specified alone, a full listing is produced with two parts: a copy of the input file and a listing that depends on the type of input.

If the LO parameter is omitted, a short listing is produced containing: translated and added lines, lines requiring manual action, and Conversion Aid messages and error diagnostics.

MC
MC=$char$

master control character. The special characters +, -, /, and = must be specified with the delimiter $; the special characters are master control characters used in UPDATE/MODIFY.

If MC=$char$ is specified, it is assumed that the master control character is char.

If MC is specified alone or omitted, it is assumed that the master control character is *.

MD

machine dependent usages. If MD is specified, issue a manual change message for each statement that contains at lease one of the following machine-dependent constructs:

1. Hollerith data
2. Shifts and masks
3. Intrinsic functions dealing with bit manipulation: XOR, OR, AND, COMPL, and LOCF
4. Octal and hexadecimal data
5. ENCODE and DECODE statements
6. BUFFER IN and BUFFER OUT statements

If MD is omitted, the machine-dependent constructs are ignored.

P
P=0
P=outlfn

source output file. If P=outlfn is specified, source output is written on the file named outlfn.

If P is specified alone, it is assumed that source output is written on the file named PUNCH.

If P=0 or is omitted, no source output is produced.

PD
PD=6
PD=8

print density. If PD is specified alone or if PD=8, a listing with a print density of eight lines per inch is produced.

If PD=6 is specified or omitted, a listing with a print density of six lines per inch is produced.

PO
PO=opt

source output options. If the parameter P is omitted or P=0 is specified, F45 ignores the PO parameter and does not produce a source output file. opt may be one of the following:

F — Produces a full source output file. Except for lines requiring manual action, this output is suitable for direct input to the compiler.

M — Produces a modification file. If the input is standard or sequenced, the Conversion Aid issues an error message and produced no output. If PO is specified alone, the M option applies.

S — Produces a short source output file. The output is intended for manually updating the original source decks. This output is not suited for direct input to the compiler. If PO is omitted, this option applies.

SO
SO=0
SO=n1/n2
SO=n1/n2/n3

sequenced output. This parameter, in conjunction with the input sequencing type, determines the type of sequenced that is output.

If SO=n1/n2/n3 is specified, n1, n2, and n3 must be specified. If SO=n1/n2 is specified, output is sequenced in columns 73-80 (EDITOR style), starting at n1, incrementing by n2. The greatest number of digits that field n1 can have is five or a value of 99999. If 99999 is exceeded in a F45, the next sequence number output will begin from zero.

If SO=0 , unsequenced output files are created.

If SO is specified alone, sequenced output files are created, with the sequence numbers determined by the input file. But SO defaults to EDITOR (SO=100/10) if no sequencing is supplied on input.

If SO is omitted, the mode of the output file is determined from the mode of the input file. If the input file is sequenced, omitting the SO parameter is equivalent to specifying SO, and the output files are sequenced. If the input file unsequenced omitting the SO parameter is equivalent to specifying SO=0, and the output files are unsequenced.

Note: Do not mix EDITOR sequencing (columns 73-80) with standard sequencing (columns 1-5). The results are unpredictable.

For more information, see the *FORTRAN Extended Version 4 to FORTRAN Version 5 Conversion Aid Program Version 1 Reference Manual.*

7.16.8 MNF

MNF is a FORTRAN compiler developed at the University of Minnesota. MNF is especially suited to student users, since its major features are performing extensive error checking and generating clear diagnostics. MNF should be used only for small programs or for debugging larger programs; it should not be used for production-type programs, since FTN (see Section 7.16.5) is much better suited for large production-type programs.

The MNF control statement has the following form:

```
MNF[,optional parameters].
```

If any errors are found on the MNF control statement, the following message will be printed in the user's dayfile, preceding the printed control statement:

```
ILLEGAL CONTROL PARAMETER x
```

(where x is the illegal character on the control statement). The character is ignored and the rest of the control statement is scanned. The legal parameter characters are explained below. Further documentation is found in the *User's Guide Supplement: MNF,* and the *MNF Reference Manual.*

B=binlfn — binary object file. If B=binlfn is specified, generated binary object code is written on file binlfn, which is not rewound before or after compilation. binlfn cannot be the same as the source input file (I=inlfn) or the source listing file (L=listlfn). B omitted or B alone assumes the file LGO. Loading and execution do not begin unless the user causes it with another control statement (e.g. 'LGO.') or the G parameter.

D
D=char

debugging mode. If D is given, comment source cards with C$ in the first two columns are treated as normal FORTRAN statements; the C$ is ignored. See Section 1.1 of the *MNF Reference Manual*. In addition, D causes the compiler to begin execution even if FATAL-TO-EXECUTION errors are detected during compilation. The program will run until normal termination or until the location of a FATAL-TO-EXECUTION error is reached, causing termination.

If D=char is specified, char replaces $ in the recognition of C$-type statements. Any letter, digit, or special character may be used. If D is omitted, cards with C$ in the first two columns are considered normal comment cards. If FATAL-TO-EXECUTION errors are detected, the program is not executed.

E=errlev

error level of diagnostics. Each level of message is assigned a digit value n; 1≤n≤5. If E=n is specified, all error messages of level n and lower are not output. The levels are:

1 COMMENT
2 NOTE (NON-STANDARD, i.e. NON-ANSI). This is the default.
3 CAUTION
4 WARNING
5 FATAL-TO-EXECUTION

If E=0 is used, all six levels of error messages (COMMENT, NON-STANDARD, CAUTION, WARNING, FATAL-TO-EXECUTION, and DEADLY-TO-COMPILATION) will go to the source listing file following the source statement causing such a message.

The level DEADLY-TO-COMPILATION is not assigned a value since only two messages may occur at this level: POSSIBLE MACHINE ERROR and PROGRAM EXCEEDS STORAGE. When the latter message occurs, all current generated binary code is cleared and the compiler continues compiling the remaining FORTRAN statements and checks for compilation errors. See Appendix G of the *MNF Reference Manual* for a further explanation of the error types.

Since the MNF compiler tries to identify all errors or possible errors, it will sometimes give out COMMENT and CAUTION messages for correct code. In this case, E=3 specified in later runs is useful in avoiding those messages. If E=5 is specified, the program will be executed even if FATAL-TO-EXECUTION errors are detected since no error messages are printed.

F

forced stores. If F is used, stores are forced before loads, and operands that are formal parameters or in EQUIVALENCE or COMMON statements are not remembered across statements. If F is omitted, MNF assumes that each computer word is known by a unique name within the program and each subprogram. Under certain circumstances, it is possible for a single computer word to have more than one variable name associated with it. If so, there are a few cases in which the wrong answer will be generated when the MNF compiler tries to optimize by using operands already in the machine operating registers that are formal parameters or in EQUIVALENCE or COMMON statements.

Most programs will run correctly without the F parameter, but if it is used and the output results differ from the normal mode then the program has a computer word known by two names.

G — go load and execute after compilation. If G is used, loading and execution occur immediately after compilation, without need for further control statements.

I=inlfn — Source input file. Source code to be compiled appears on file inlfn. If I is specified alone or omitted, the file INPUT is assumed. The input file cannot be the same as the binary output or source listing files (see L and B parameters).

J — printer carriage control between program units. If J is omitted, a "T" carriage control is used between program units on the source listing. This causes a page eject and sets print density to 8 lines per inch on central site printers (source "B"); and acts as a single-space on all other printers.

K — time-sharing version of MNF. If K is used, a "time-sharing" version of the MNF compiler is used. It allows a free-field input format, described in the *MNF Reference Manual*; prohibits the printing of a reference map; and implies selection of the following parameters: D,L=0,R=0,E=2,O=0. Refer to the *MNF Reference Manual* for details.

L=listlfn
L=0 — source input, error messages and reference maps listed on file listlfn (unless the O parameter conflicts—see below). If listlfn is not specified, or if the L parameter is omitted, the file OUTPUT is assumed. L=0 inhibits all listable output controlled by the L, O, and R parameters. Error messages are still listed, however.

Refer to the *MNF Reference Manual* for details of the interaction between the L parameter and the MNF source statements PAGE, LIST, and NOLIST.

O=outlfn
O=0 — pseudo-COMPASS object code listing. If O=outlfn is specified, the pseudo-COMPASS object code is listed on local file outlfn. If O is specified alone, each FORTRAN statement on the source listing is followed by a pseudo-COMPASS listing of the object code generated by it. If O is omitted or O=0 is given, the mnemonic pseudo-COMPASS object code listing is not produced.

If the L and O parameters specify different local file names, all listable output (source statements, pseudo-COMPASS, error messages, and reference maps) is directed to the last outlfn specified on the MNF control statement.

Refer to the *MNF Reference Manual* for details of the interaction between the O parameter and the MNF source statements LIST, NOLIST, CODE, and NOCODE.

P=n — line count limit. If P=n (a decimal number) is given, the execution line count for the standard OUTPUT file is set to n. TRACE messages are counted toward the line count limit. If P is specified alone, the execution line count is set to zero. Thus, the first output on the standard OUTPUT file will terminate the the job. If P is omitted, the execution line count limit for the standard file OUTPUT is set to 5000 lines. (There is no limit to the number of lines used during compilation, or during execution if the file is not named OUTPUT.)

R=n — cross reference listing. n is one of 0, 1, 2, 3, 5, 7.

If R=0 is used, no cross reference listing is written on the OUTPUT file. This is the default.

If R=1 is used, cross references are written on the source listing file for FORTRAN statement numbers and names sorted in numerical and alphabetic order except for unused variables having no active references in the current subprogram.

If R=2 is used, a cross reference map for variables and arrays sorted by octal address is written on the source listing file. This is helpful in interpreting the octal dump of an abnormal termination.

R=3 means a combination of R=1 and R=2.

R=5 means the same as R=1 but includes unused variables (nulls) having no execution reference in the current routine; and references to invented subroutine names in the MNF library.

R=7 is a combination of R=2 and R=5; this is the most complete map.

If the cross reference listing parameter is omitted (default value) the value R=0 is assumed and no cross reference map is written. Refer to the *MNF Reference Manual* for details of the interaction between the R parameter and the MNF source statements LIST, NOLIST, REFERENCES, AND NOREFERENCES.

T trace possible errors. If T is specified, most of the tracing facilities of MNF are turned on at the first statement of the program. These only cause printed output if an error occurs and give total usage counts at the end of execution. They are:

TRACE DOLOOPING
TRACE FORMATION
TRACE SUBSCRIPTS
TRACE TRANSFERS
TRACE STATEMENT NUMBERS
TRACE SUBPROGRAM CALLS

See Section 10.1 of the *MNF Reference Manual* for a more complete explanation of these TRACE statements.

If T is omitted, normal object execution code is produced for FORTRAN statements.

Note: TRACE output counts toward the execution-time line limit on the file OUTPUT. Refer to the P parameter, above, to change this limit.

Y bypass degenerate loops. If Y is specified, degenerate loops will not be executed at all. Thus, DO loops and implied DO loops will be completely bypassed if the value of the initial parameter is greater than the value of the terminal parameter, while FOR loops will be completely bypassed if the value of the initial parameter is less than the value of the terminal parameter. If Y is omitted, DO loops, FOR loops and implied DO loops will always be executed at least once no matter what the values of the initial and terminal parameters.

Z zero undefined values. If Z is used, undefined variables or arrays are initially set to zero. If Z is omitted, undefined variables or arrays will be initially set to a negative indefinite value.

Examples: For examples and further documentation of MNF, see the *MNF Reference Manual* and the *User's Guide Supplement: MNF.*

Out-of-order in original manual
Oct 2009?

7.17 On-Line Documentation

The utility described in this section allows the user to obtain directly from the system documentation of products, features and services.

7.17.1 HELP

The HELP statement may be used to obtain:

1. announcements of new software, current problems, bug fixes, production schedules and policies.

2. descriptions of any SCOPE/HUSTLER control statement, special interactive command, Front-end command, or EDITOR directive.

3. descriptions for any subprogram, programming package, or data file available from the HUSTLER Auxiliary Library.

4. descriptions for entries on User Auxiliary Libraries.

Each announcement or description is a separate entry in the HELP information file. Information can be requested for individual entries, for entry categories, or for entries revised or created after a specified date. For a description of entry categories and their codes, see Appendix A of the *HAL Reference Manual*, or Appendix E of the *Interactive System User's Guide*.

Information displayed may include a title, abstract, main body of a description, job structure, and other entry information.

The HELP control statement is as follows:

```
HELP[,optional parameters].
```

Most information is supplied by optional parameters on the HELP statement. These are of two types: file parameters and retrieval parameters. The parameters may be specified in any order.

If the form 'HELP.' or 'HELP,file-parameters-only.' is used, the user will receive information about the HELP utility itself.

File Parameters

File parameters describe input and output files and their attributes.

CC
NOCC — When CC is specified, HELP will insert carriage controls between each description output to provide spacing. These may be suppressed by the parameter NOCC. For connected output files, the default is NOCC. In all other cases the default is CC.

CY=nn — specifies the user auxiliary library file cycle number.

L*usrlib — specifies that a User Auxiliary Library is to be used rather than the standard HUSTLER Auxiliary Library.

O=outlfn
O=0. directs HELP description output to local file outlfn. The batch default is OUTPUT; the interactive default is TTYTTY. If O=0 is specified, output will be suppressed.

OC=AF
OC=DC specifies the character codes to be used for the output file. AF selects ASCII character codes and DC selects the standard CDC Display character codes. The default output code is AF.

PD=6
PD=8 used to specify print density for batch line printers. 6 selects six lines per inch, 8 selects eight lines per inch. The default is 8.

PW=rdpw supplies a read password, if needed, which is used to attach the User Auxiliary Library HELP file.

S=srclfn
S=0 writes out a source file of the descriptions. S=0 suppresses the source file; this is the default.

Retrieval Parameters

Up to 30 retrieval requests may be specified in one call to HELP. In batch, if one card proves insufficient, the parameters may be continued on subsequent cards; the preceding card should terminate with a comma.

Each retrieval request of the form

```
[moddate=][CAT=][ND*][prefix*]keywrd
```

and is separated from other parameters by a comma.

moddate= causes printing of only those descriptions that have been modified since moddate. Since moddate is of the form: 'mm/dd/yy=' where 01 ≤ mm ≤ 12, dd is a legal day within the specified month, and 00 ≤ yy ≤ 99. For example: 01/12/76 indicates January 12, 1976.

CAT= is used to output all entries of a category or to list selected entries on the library category order. If 'CAT=' is used, 'keywrd' may not be an entry name.

ND* suppresses the directory (header title) when used in conjunction with the keyword ALL. With any other keyword directory suppression is the default condition.

prefix* is any permutation of the following single letter mnemonics:

T	title
A	abstract
B	body
D	deck structure
E	entry information
F	full description (includes T,A,B,D,E)

If * is specified with no prefix, a prefix of T is assumed.

If 'prefix*' is not specified from batch or with 'O=outlfn' in an interactive job, a prefix of F is assumed.

keywrd may be any of the following:

a legal entry name on the HUSTLER Auxiliary Library or User Auxiliary Library
a legal HUSTLER Auxiliary Library category
ALL (includes every entry in the library)
PROGRAM (only HELP entries for main programs)
SUB (only HELP entries for subprograms)
DATA (only HELP entries for data files)
HELPONLY (only HELP-only entries)
DOC (only those entries having a HELP description)
NODOC (only those entries having no HELP description)
ERROR (explains the last error condition that occurred)

If 'keywrd' is any of the special names such as ALL, PROGRAM, DOC, etc. or a category name, and 'CAT=' is not specified, an existing library entry with that name will override and be listed instead of all routines in the category.

Note that the optional parts of a retrieval request are order-dependent. Thus, 'moddate=', if specified, must be first, 'CAT=' must be second, etc.

Examples:

```
HELP,SCHEDULE.
```

This prints a copy of the production schedule.

```
HELP,F*DISPOSE.
```

Prints a full description of the DISPOSE card format and how the utility is used.

```
HELP,A*ED,T*F1.
```

Prints abstracts of all EDITOR directives (category *ED) and titles for all entries in category Fl (matrix manipulation). This is the most commonly used form.

```
PNC
job card
HELP,0=A,04/01/79=F*ALL.
DISPOSE,A,PAF.
6/7/8/9
```

Prints full descriptions of all entries modified since April 1, 1979; information is output on file A, then disposed to an upper/lower case line printer at the central site.

Complete documentation for HELP can be found in the *HAL Reference Manual*, Section 3.1.

7.18 Cyber Record Manager Utilities

Cyber Record Manager (CRM) is a group of routines that provide an interface between user programs and the system routines that read and write files on hardware devices. The control statements in this section call CRM utilities for manipulating indexed-sequential files. Users are referred to the CDC *Cyber Record Manager Reference Manual* (Publication No. 60307300) for more information.

7.18.1 ESTMATE

The ESTMATE utility aids in calculating indexed block size, data block size, and buffer size for indexed-sequential files. Directives are read from local file INPUT, and results are printed on local file OUTPUT. The control statement is:

```
ESTMATE[,optional parameters].
```

NR=nrec file size; number of records. Default is 100000.

KS=keysize length of key in characters; must be specified as 10 for floating point and 5 or 10 for integer keys.

MR=maxrecsz maximum record size in characters. Default is 1000.

MI=minrecsz minimum record size in characters. Default is 500.

Complete documentation may be found in the *Cyber Record Manager Reference Manual*, Chapter 6.

7.18.2 IXGEN

IXGEN creates an index file for alternate key access of an existing indexed-sequential (IS), direct access (DA), or actual-key (AK) file.

```
IXGEN,prilfn[,dirlfn].
```

prilfn local file name of an existing indexed sequential (IS) direct access (DA), or actual-key (AK) file.

dirlfn Local file name of the file containing the necessary RMKDEF directives. The default is INPUT.

For details concerning the use of IXGEN, consult the *Cyber Record Manager Reference Manual*, Chapter 9.

7.18.3 SISTAT

SISTAT prints statistical information about an indexed-sequential file.

```
SISTAT,islfn[,outlfn].
```

islfn local file name of indexed-sequential file.

outlfn optional local file name on which to print the statistics. The default is OUTPUT.

Consult the *Cyber Record Manager Reference Manual,* Chapter 6, for complete documentation.

7.19 Data Base Management Utilities

DMS-170 is a data base management system which has the means to:

1. Create a common-user data base.
2. Provide and maintain a variety of data structures for specific users.
3. Control, monitor, and interpret data base access requests from applications programs.
4. Establish file integrity and security.
5. Guarantee maximum efficiency for accessing, modifying, and processing data base information.

The control statements in this section access utilities which perform these, and other, data base management functions.

7.19.1 DDL

DDL (Data Description Language) is a high-level language that parallels the COBOL language convention of grouping English terms into sentence-type statements. DDL statements are used to describe the characteristics and structure of individual items in the data base in two distinct, but related, formats: the schema, which describes the entire data base, and the sub-schema, which describes the portion of the data base accessible to one or more applications programs.

The DDL control statement to provide the DDL compiler with information about the schema is:

```
    DDL[,optional parameters].
```

DS specifies that a DDL schema is to be compiled.

I=inlfn the local file name of the source input file. The default is INPUT.

L=listlfn
L=0 the local file name of the source listing file. The default is OUTPUT. If L=0 is specified, only diagnostics and associated statements are listed.

NI=n specifies the number of 10-word blocks allocated for the schema record buffer. The default is one-fourth of the available memory.

SC=schlfn specifies the local file name of the schema; the default is the first 7 characters of the schema name in the schema description entry of the source program.

Further documentation on the schema is found in the CDC *DDL Reference Manual,* Volume 1 (Publication No. 60498400).

The following DDL control statement provides the DDL compiler with information related to a specific COBOL sub-schema. It is used to add, replace or delete a sub-schema in the sub-schema library.

```
DDL,CB,SB=sublfn[,optional parameters].
```

CB specifies a COBOL/DDL sub-schema for use with a COBOL 4 application.

SB=sublfn specifies the local file name of the sub-schema library.

A produces a list of sub-schemas and their corresponding schemas, together with their creation dates, from the library indicated by the SB parameter.

I=inlfn specifies the name of the source input file; the default is INPUT.

L=listlfn
L=0 specifies the name of the output file onto which the source listing is written; the default is OUTPUT. If L=0 is specified, only diagnostics and associated statements are listed.

N indicates that the sub-schema is to be compiled but not added to the sub-schema library.

P the specified sub-schemas are purged from the sub-schema library; no compilation takes place.

R replaces the existing sub-schema in the sub-schema library with the sub-schema compiled from the source program on the input file. Replacement takes place only if no compilation errors other than informative diagnostics are encountered.

SC=schlfn specifies the local file name of the schema; this parameter overrides the schema name in the Title Division of the source program. The default is the first seven characters of the name specified in the program.

More complete documentation can be found in the CDC *DDL Reference Manual*, Volume 2 (Publication No. 60498500).

The following control statement provides the DDL compiler with information related to the creation of a QUERY-UPDATE/DDL sub-schema directory.

```
DDL,QD,SB=sublfn[,I=inlfn][,L=listlfn].
```

QD specifies a QUERY-UPDATE sub-schema directory is to be produced.

SB=sublfn specifies the local file name of the sub-schema.

I=inlfn specifies the local file name of the source input file; the default is INPUT.

L=listlfn
L=0 specifies the local file name of the source listing output file; the default is OUTPUT. If L=0 is specified, only diagnostics and associated statements are listed.

Complete documentation can be found in the CDC *DDL Reference Manual*, Volume 3 (Publication No. 60498600).

7.19.2
DFRCV and DFRST

The Data Base Utilities (DBU) consist of a utility to recover a data base (DFRCV) from a given point and a utility to restore a data base (DFRST).

DFRCV is initiated to recreate a lost or partially destroyed data base from a record (called a log) of changes and a previously dumped backup copy of the data base.

DFRST uses a log of the data base before changes are made to remedy the problem of invalid data in the data base.

The control statements for these utilities are as follows:

```
DFRCV,L=loglfn[,optional parameters].

DFRST,L=loglfn[,optional parameters].
```

The L=loglfn parameters are required; all others are optional. Except where noted below, all parameters have identical meanings for both control statements.

L=loglfn

the local file name of the log file as it appears in the ATTACH or REQUEST (for tape) control statement used in the recover or restore control statement setup. A name having 1 to 7 characters is permissible. If no other parameters are specified, the entire contents of the log file are used to perform the recovery or restoration operation. If multiple log files exist, the recover or restore job must be-run for each log file.

A=areaid

the 1 to 7 character area identification of the area to be recovered or restored as it appears in the log file. This is the first 7 characters of the schema area name. If no area identification is specified, all areas referenced in the log file are processed in the recovery or restoration operation.

C=ckptno

An integer 1 to 2047 corresponding to the applicable checkpoint number recorded in the log file. If the checkpoint parameter is specified, it designates a termination point for the restore or recover utility. This point must be within the startdate and starttime and enddate and endtime period of D and T are specified in the control statement.

D=startdate[/enddate]

The options startdate and enddate are expressed in the format yyddd, where the first two digits are the year, the last three, the number of the day in the year. If the D parameter is specified, the T parameter must also be included.

The range of records to be processed is determined by the range specified in the startdate and enddate values. If only one date is given in the DFRCV control statement, it is considered to be the enddate. Records on the log file containing a date later than enddate are not recovered. The default startdate is the beginning of the log file.

If only one date is given in the DFRST control statement, it is considered to be the startdate. Records on the log file containing a date earlier than startdate are not restored. The default enddate is the end of the log file.

The combination of startdate and starttime must always be earlier than the combination of enddate and endtime.

H=ept/libname

The 1 to 7 character name of the entry point of a hash routine. The H parameter applies to direct access (DA) files and is used only if a hash routine is supplied by the user. The hash routine is stored in a library; the library name must be included as shown in the format.

If no user hash routine is specified, the default is a hash routine supplied by the system. The hash routine specified applies to all DA files processed in the recover or restore operation. If the hash routine is stored in a user library, this library must be attached with a local file name specified as the hash library name.

O=outlfn

the output file to which nonfatal error diagnostics are written. The default is the file OUTPUT.

T=starttime[/endtime]

expressions used in the format 'hhmmss', designating hours, minutes, and seconds. If the T parameter is specified, then the D parameter must also be included.

The startdate and starttime and endate and endtime sequences specified in the D and T parameters determine the range of records to be processed. For the DFRCV control statement, the default starttime is beginning of the log file. The default endtime is the end of the log file for the DFRST control statement.

The combination of startdate and starttime must be earlier than the combination of enddate and endtime.

U=userid

the 1 to 10 character user identification contained in the log file. This is the COBOL program name. If no user identification is specified, the default is all users. If the user identification contains a hyphen, the entire name must be enclosed in dollar signs ($). Both the U and A options serve to restrict the scope of recovery or restoration to specific log entries.

Complete documentation of these utilities is found in the CDC *Data Base Utilities Reference Manual* (Publication No. 60388900).

7.19.3 QU

QUERY-UPDATE (QU) is a high level, English-like conversational language for use in querying and manipulating data files organized under Cyber Record Manager with multiple indexing. It is designed for use in an interactive environment, though usable in batch mode as well. QU features simple syntax, the use of arithmetic and Boolean expressions, callable procedure sessions that can be recorded in a catalog, and an extensive report writing facility.

QU is called by the following control statement:

```
QU[,optional parameters].
```

I=inlfn specifies the directive input file. If I is given without inlfn, the file COMPILE is assumed.

Batch: The default file is INPUT.

Interactive: The default file is ZZZZZIN, which is connected to the terminal. Specifying the I parameter causes execution to occur as in batch mode; directives are input from the file inlfn rather than from the terminal.

O=outlfn specifies the output file.

Batch: The default is OUTPUT.

Interactive: The default is ZZZZZOU, which is connected to the terminal. Specifying outlfn causes execution to occur as in batch mode. QUERY UPDATE responses are received by the output file; responses are displayed at the terminal only if the user connects an output file when the O parameter is specified.

T=translfn specifies the transaction file, which receives a copy of all directive input and system output other than generated reports.

Batch: The transaction file is a duplicate of the output file.

Interactive: The specified file receives a copy of terminal input and QU output; at completion of terminal use, the transaction file can be printed to obtain a listing of directives entered at the terminal and system diagnostic messages.

TL=transleng specifies in characters the maximum length of a transmission. A decimal integer must be specified for transleng; it cannot exceed 6 digits. The minimum transmission length is 20 characters. 1030 is the default for both batch and interactive modes.

For further documentation see the CDC *QUERY-UPDATE Reference Manual* (Publication No. 60387100).

7.19.4 REPORT

The REPORT utility program is a part of QUERY-UPDATE that produces reports according to specifications supplied by the user via directives. The control statement is as follows:

```
REPORT,R=rptname,T=tbllfn[,optional parameters].
```

R=rptname specifies the name of the report to be prepared; this parameter must be specified.

T=tbllfn specifies the table file that contains the report format; this parameter must be specified.

I=inlfn requests a report to be prepared from data in the specified input file.

If both the I=lfn and P parameters are included, a two-page sample report with data values from the specified input file will be generated.

P requests a two-page sample report with dummy data values.

V=vallfn specifies the file that supplies names and values of temporary data items to be preset before report generation begins. If V is given alone, the file INPUT is assumed.

Further documentation is provided in the CDC *QUERY-UPDATE Reference Manual* (Publication No. 60387100), Chapter 6.

7.20 Program Packages

The control statements in this section provide access to program packages available to users of the MSU 6000 system.

7.20.1 APEX

APEX III is a software package designed to solve linear programming problems. Basically, it is an optimization system whose main function is to optimize MPS (an industry standard) formatted linear models to either maximize gains or minimize losses. APEX users can solve linear programming problems in one of three ways:

1. submit a single control statement which specifies the program options to be employed on the problem,

2. prepare an APEX control program that consists of several control statements and more explicitly indicates how the problem should be solved, and

3. utilize APEX verbs as callable subroutines.

Each of these approaches satisfies a different set of user needs. Normally, users employ a single control statement if they want to exercise a relatively small number of options, since all options must appear on one card. If more options are needed, then the user must prepare a control program.

The single control statement method is briefly described below. Further documentation is available in the CDC *APEX-III Reference Manual* (Publication No. 76070000).

APEX is called by the following control statement:

```
HAL,APEX,SOLVE[=datalfn],drctn[,optional parameters].
```

SOLVE=datalfn specifies the local file that contains the program data. INPUT cannot be the file specified. If SOLVE is given alone, the file TAPE1 is assumed. 'SOLVE' may be abbreviated 'S'. This parameter is required.

drctn indicates the direction in which optimization is to be performed. Legal values are:

MIN minimizes the objective function.
MAX maximizes the objective function.

This parameter is required.

ABM
ABT causes execution of ABORT rather than EXIT when major, minor, unfeasible, nonoptimal exit conditions exist.

BCD=mpslfn
BCDV=mpslfn causes APEX to write the current model out in MPS format. If BCD or BCDV are given alone, the file TAPE13 is assumed.

BND=bndname — provides a bounds set with a user-specified name.

CM,
MS — causes the internal arrays that store vector names, bounds, and ranges sets information to be stored in either central memory (CM) or mass storage (MS).

DEV — indicates that APEX should use the DELVEX verb rather than the PRIMAL verb for solving the problem.

DT — causes APEX to print detailed inversion statistics.

EQ — specifies that APEX should produce a full equation listing for the linear programming model.

GO — causes APEX to do a full output when the problem is feasible.

IG — causes APEX to continue processing even though it encounters minor input, basis and/or revise errors.

INB=inlfn — instructs APEX to access a user-supplied file to obtain a starting basis for this computer run. If INB is specified alone, the file TAPE3 is assumed.

INU — instructs APEX to process the problem by starting with a unit basis.

L
LB
LC
LR — indicates that APEX should produce a list of rows, columns (LC), bounds (LB) or ranges (LR) for the problem; 'L' gives a complete listing.

LOG=val — indicates how often an iteration log line should be printed; 'val' is the number of iterations between log lines. The default is 10,000.

MCH — establishes iteration frequency of monotonicity check in iterating routines. If no gain in objective function value or decrease in number of unfeasibilities after LFMONOT iterations; PRIMAL exits with LBRANCH=3. The standard value of LFMONOT is 200.

MIP — causes APEX to use the MIXINT algorithm for problem solution. If this parameter is not present, APEX will solve the continuous problem via the PRIMAL algorithm (or DEVEX algorithm if the DEV parameter is present).

N=probname — specifies what name will appear on the NAME card in the MPS-formatted model.

NTS=k — optional parameter allowed only when the MIP parameter is used; causes termination of MIXINT processing when the kth integer solution is found.

OBJ=objname — provides the objective function with a user-specified name.

OPT — optional parameter allowed only when the MIP parameter is used. If present, MIXINT searches for the optimal integer solution. If not present, MIXINT will terminate upon finding an integer solution within ten percent of optimal.

PT or SP or
FS or PS or
NO or PD or
PPD or FPD
and/or O=outlfn — indicates how much and what type of solution output should be produced by this processing run (see page 2-11 of the *APEX III Reference Manual* for a description of each type).

R
R=revlfn — causes APEX to revise the input model; revisions are on file revlfn. If R is specified alone, the file TAPE14 is assumed. 'RDN=name', specifying the revise deck name, and 'RPS', specifying that only header cards should be printed, are optional parameters that may be given with the R parameter.

RANGE — indicates that APEX will perform an activity range of the optimal solution rather than the standard report.

RHS=rhsname — provides the right-hand side with a user-specified name.

RL=val — indicates that APEX will terminate execution when it has used the user-specified number of resource units. A resource unit is a term used for system accounting information. This does not override the limits specified on the job card or the APEX ARULIM card.

RNG=rngname — provides a ranges set with a user-specified name.

SOF=soflfn — changes the standard system OUTPUT file to file soflfn.

TER=valu — indicates that APEX will terminate execution when it has performed a specified number of iterations.

TRI=trilfn — optional parameter allowed only with MIP; instructs APEX to access a user-supplied file to TREEIN data.

TRO=trolfn — optional parameter allowed only with MIP; instructs APEX to do a TREEOUT after solving the MIP problem.

YP or NP — specifies the perturbation strategy that APEX should use during the optimization process.

YSB=solblfn — instructs APEX to save the solution basis for this problem on file solblfn. If YSB is given alone, file TAPE4 is assumed.

If a control program is written, the following APEX control statement will be used:

```
HAL,APEX[,C=contlfn].
```

C=contlfn — file containing the control program. The default is INPUT.

Complete documentation of each method of APEX processing is found in the CDC *APEX-III Reference Manual* (Publication No. 76070000).

7.20.2 GCS

The Graphics Compatibility System (GCS) is a FORTRAN-based graphics system designed for use on a wide variety of graphics devices. The control statement for GCS is:

```
HAL,GCS,DEVICE=devtype[,optional parameters].
```

DEVICE=devtype

a required parameter specifying the graphics device type used for the current GCS job; devtype may be one of the following:

TTY
CAL or CALCOMP
PR or PTR or PRINTER
TEK or 4010 or 4012
TK4 or 4014
GSI

'DEVICE' may be abbreviated 'DEV'.

I=inlfn

specifies the input file. For device types TTY, TEK, TK4 and GSI, the default is the connected file ZZZZGIN; for PR and CAL, the default is file INPUT.

LIMIT=maxsize

the maximum amount of plotter paper that can be used, in inches. The default is 1200 inches. 'LIMIT' may be abbreviated 'L'.

O=outlfn

specifies the output file. For device types TTY and GSI, the default is the connected file ZZZZGOT; for TEK and TK4 the default is the connected file TEKOUT; for PR, the default is the file OUTPUT; for CAL, the plot is written directly on the system plot file.

OFF

means eliminate GCSLIB from the global library set. This parameter is most useful when no relocatable loading is required (as in the case of an already-loaded overlayed program), and the only action requested is the creation of the file GCSDATA.

PAPER=pwidth[/ptype]

describes the size and type of paper desired for the GCS plot, if other than the normal plot queues (see TYPE). Legal values are:

pwidth	ptype
15 or NARROW	TRAN or TRANS
35 or WIDE	MY or MYLAR
	TRI or TRIACETATE
	RAG or RAGBOND
	VEL or VELLUM

The default pwidth/ptype is NARROW/TRANS.

PC=plotcost
specifies the maximum cost of the plot in cents. The default is no limit.

PENpnum=pentype[/pencolor]
specifies the pen number, type and size, if other than the normal plot queues (see TYPE). The value pnum may be 1, 2, or 3; there is no default. Legal values for pentype and pencolor are:

pentype	pencolor
BP	BK or BLA or BLK or BLACK
2	R or RED
3	BLU or BLUE
4	G or GRE or GRN or GREEN
5	Y or YEL or YELLOW
6	
8 or WIDE	

The defaults are: PEN1=BP/BK; PEN2=BP/R; PEN3=BP/BLU.

PT=plottime
specifies the maximum plot time in seconds. The default is no limit.

TYPE=plottype
specifies the plot queue to which the GCS plot is to be routed. Legal values for plottype are:

0 or STDBP
1 or BIGBP
5 or STDINK
6 or BIGINK
10 or COLLATED

The default is 0. See the *Plotting and Graphics Reference Manual*, Table 2.2 for a description of the plot queues.

GCS is fully documented in the *Introduction to the Graphics Compatibility System*, published by the U. S. Military Academy.

7.20.3 SPSS

The Statistical Package for the Social Sciences (SPSS) is an integrated system of programs which combines a comprehensive set of procedures for data transformation and file manipulation with a large number of statistical routines. Complete documentation of SPSS is available in the *Statistical Package for the Social Sciences* published by McGraw-Hill Company, and the Computer Laboratory publication *SPSS-6000 Supplement*.

SPSS is called with the following control statement:

```
HAL,SPSS[,optional parameters].
```

A=altlfn Alternate BCD output is to be written on file altlfn (matrices, residuals, Z scores, etc.). The default altlfn is BCDOUT.

BL=kkk Input data file is a stranger tape (S or L format) containing fixed length records (see RL) with blocks of maximum size kkk characters. RL must be specified for both S and L tapes. BL must be specified for L tapes but is optional for S tapes (kkk less than 5120). If BL is omitted, the input data file is assumed not to be an S or L tape.

D=datalfn SPSS input data is read from file datalfn. The default is INPUT.

ER=nnn Maximum number of allowable I/O errors (illegal characters in 'F' format fields) or illegal computations is nnn. The default for nnn is 100.

G=syslfn SPSS system file will be read from file syslfn (i.e. due to GETFILE). The default syslfn is GTFILE.

G1=archlfn1 Input archive file 1 is archlfn1. The default archlfn is GTARC1.

G2=archlfn2 Input archive file 2 is archlfn2. The default archlfn is GTARC2.

G3=archlfn3 Input archive file 3 is archlfn3. The default archlfn is GTARC3.

G4=archlfn4 Input archive file 4 is archlfn4. The default archlfn is GTARC4.

G5=archlfn5 Input archive file 5 is archlfn5. The default archlfn is GTARC5.

I=inlfn SPSS control cards compose the next section on file inlfn. The default inlfn is INPUT.

L=listlfn
O=outlfn SPSS printed output will be written on file listlfn or file outlfn. The default is OUTPUT.

L6 Printed output will be formatted at 6 line-per-inch printing. The default format is 8 line-per-inch printing.

OS=oslfn File oslfn contains an OSIRIS dictionary for use in conjunction with the OSIRIS VARS card. A binary tape file or disk file may be used. RL must also be specified.

RL=nnn Input data file is an S or L tape (written in BCD mode) containing fixed length records of length nnn characters. If RL is omitted, the input data file is assumed not to be an S or L tape.

S=svlfn SPSS system file will be written on file svlfn (i.e., due to SAVE FILE or SAVE ARCHIVE. The default svlfn is SVFILE.

X=binlfn
X=0 File binlfn will be used to store the input data in binary form for second and subsequent procedure calls. The default binlfn is XXSS2. If X=0 is specified, no binary file will be written. (This option can save time and disk space on runs that execute only one procedure.)

7.20.4 STAT

The MSU STAT system is a package of statistical programs and subprograms, written at MSU. Data transformations with STAT require a knowledge of the FORTRAN language and some ability to use subprograms. The STAT control statement is

```
HAL,STAT4.
```

There are no parameters.

Complete documentation of the MSU STAT System is available in the *MSU STAT System User's Guide* and the User's Guide Supplement: *User-Supplied Subroutines for the MSU STAT System.*

8
Extensions of CDC COMPASS

Several sections in the original version of this chapter (entitled "Extensions and Clarifications of Software Products") have been removed in Revision F because the material in those sections is either documented more fully in other Computer Laboratory publications, or is obsolete.

Section 8.1, "Maintaining Compatibility with Future Improvements" (pages 8-1 through 8-4), is obsolete.

Section 8.2, "FTN" (pages 8-5 through 8-16), has been expanded and reprinted in the *User's Guide Supplement: FORTRAN Extended Library Routines*.

Section 8.3, "RUN" (page 8-16), is obsolete.

Section 8.4, "RUNT" (pages 8-17 through 8-21), is obsolete.

Parts of Section 8.5, "COMPASS" (pages 8-25 through 8-44), have been retained, since they describe some COMPASS macros that are not documented in any other publication. Pages 8-22 through 8-24 are deleted as obsolete.

Parts of Section 8.6, "COBOL" (pages 8-45 and 8-46), are obsolete. The entire section has been removed and will be rewritten and available in another document.

Section 8.7, "SIS" (page 8-46), is obsolete.

Section 8.8, "APL" (page 8-46), is obsolete.

Section 8.9, "SNOBOL" (pages 8-46 through 8-53), is obsolete.

Numerous pages missing here; they were removed from the publication prior to this revision.

8.5.4 AUTOMATIC FIELD LENGTH ADJUSTMENT

The amount of central memory needed to assemble a COMPASS program depends on the size of the program, the macros used, the system text used, and other variables. But unlike other "size-sensitive" system programs, COMPASS does not require AUTORFL(PART) for assemblies that require more than the default field length. When AUTORFL is ON, COMPASS will load at the default field length (currently 45000B) and then automatically request additional core as needed, up to the maximum field length specified by the job card CMfl parameter.

8.5.5 PERMANENT FILE MACROS

Users who perform permanent file functions through COMPASS macros should note the following additions:

> Two additional options are available for attach and catalog functions.
>
> Several new error codes may be returned when the RC option is requested.

The major part of this section, however, is devoted to a detailed description of the file definition block, since this information is no longer available in the SCOPE 3.2 Reference Manual.

Permanent File Functions

With the aid of the system macros described in this section, the COMPASS programmer can catalog, attach, extend, and purge permanent files through object-time requests just as easily as through control card requests, and with greater flexibility. The first step in making an object-time permanent file request is to construct a table, called a *file definition block* (FDB), containing the parameters necessary to execute the desired function. If he wishes, the user can instruct the system to return an error code to this table if the request cannot be carried out. An unsuccessful control card request, on the other hand, will always cause job termination.

The FDB macro and the format of the file definition block are described in the second part of this section. The following macro calls are used to issue the attach, catalog, extend, and purge requests.

```
CATALOG   fdbaddr,RC
ATTACH    fdbaddr,RC
PURGE     fdbaddr,RC
EXTEND    fdbaddr,RC
```

fdbaddr the location symbol associated with the FDB macro, i.e., the address of the fifth word of the file definition block.

RC an optional parameter which requests that a return code be stored in bits 9-17 of location fdbaddr.

The permanent file macros use the FDB in the same way that the file action macros (e.g., READ, WRITE, SKIPF) use the FET. They store a function code in bits 0-8 of location fdbaddr and then format a request to one of the PP routines (PFA, PFC, PFE, or PFP) by way of Central Program Control (CPC):

<table>
<tr><td>59</td><td>41</td><td></td><td></td><td></td><td>29</td><td>23</td><td>17 0</td></tr>
<tr><td colspan="5"></td><td colspan="2">RJ</td><td>CPC</td></tr>
<tr><td>PFx (x=A/C/E/P)</td><td>1</td><td>1</td><td></td><td colspan="2">0000</td><td></td><td>fdbaddr</td></tr>
</table>

The function codes that are masked into fdbaddr are listed following the description of the file definition block. There, the reader will note a column of codes headed "RT (RC implied)". The RT (real time) parameter is not recognized in any of the macro calls listed above, but the RT function codes are automatically placed in the FDB whenever the program is run under MISTIC2. The determination of whether the program is in batch or interactive mode is made by the system PP routines. As a result of the RT option, control will be returned to the program when a permanent file cannot be immediately attached because it is currently assigned to another job (return code 25), or when a permanent file cannot be immediately cataloged because the permanent file directory is temporarily full (return code 26). In batch mode, either condition causes the job to be swapped out until the request is fulfilled.

Two additional options, which may be identified as the "reject duplicate names" option and the "request status information" option, can now be specified by parameter entries in the file definition block, although there are no corresponding parameters recognized by the FDB macro. Methods for specifying these options will be described later.

The FDB Macro

The FDB macro constructs a file definition block.

```
          fdbaddr   FDB     lfn,pfn,params
```

fdbaddr — the location symbol that will be associated with the *fifth* word of the file definition block. Subsequent permanent file requests will refer to the FDB by this address rather than the first word of the block.

lfn — the local file name, which will be stored in L format at location fdbaddr.

pfn — the permanent file name, which will be stored in L format in the first four words of the table. When a file is cataloged, the PP routine PFC forces the permanent file name to conform to two rules:

1) PFC will convert any trailing blank characters (55B) to binary zero (00B).

2) PFC will convert any embedded binary zero characters (00B) to display code blanks (55B).

This conversion is not performed by the ATTACH routine PFA. If a permanent file name is modified when it is cataloged, it is the user's responsibility to supply the correct name when he later attempts to attach that file.

params — a list of parameters identical in format to those of an ATTACH or CATALOG control card. Consecutive parameters are separated by commas and the list is terminated by a blank. The user need specify only those parameters necessary to execute the intended functions or to obtain the desired access permissions. For example, the EXTEND and PURGE functions use only the lfn entry of the file definition block; other parameters, if present, are ignored.

Examples:

```
  PF1       FDB    TAPE1,JONESEWFILE,EX=GOOD,MD=GRIEF

  PF2       FDB    TAPE2,(F.B.I. FILE),PW=SUPER,SLEUTH,CY=5
```

```
                                                 column 72 ──┐
                                                             ↓
   PF3       FDB    OUT,(A VERY LONG PERMANENT FILE NAME),TK=VE
   ,RBOSE,RP=999,ID=PROJECT3
```

The second and third examples illustrate the use of permanent file names that cannot be specified on a control card. The third example also shows how to continue the parameter list onto an additional card.

The FDB macro generates the following table:

	59	17	8	5 0
Word 1				
2	Permanent File Name			
3	(Display Code left-justified zero-filled)			
4				
fdbaddr 5	Local File Name (left-justified zero-fill)	Return Code	Status	
6	Parameter Value (right-justified zero-filled)			nn
7	Parameter Value (right-justified zero-filled)			nn
k	zero			

Function Codes

The ATTACH, CATALOG, EXTEND, and PURGE macros set the status field of the file definition block as follows:

	RC	no RC	RT (RC implied)[1]
ATTACH	010	110	210
CATALOG	020	120	220
EXTEND	030	130	230
PURGE	040	140	240

If either the RC (return code) or RT (real time) option is specified, certain types of errors will be treated as non-fatal; if neither is specified, any error will cause abnormal termination (PP abort). The RT option cannot be

[1] Interactive programs should always check the return code for file busy status (25B) because RT codes are always selected.

specified with a macro parameter, but it is automatically selected at execution time for all MISTIC2 jobs. The RT option allows error codes 25 and 26 to be returned in place of having the job swapped out until the wait condition is cleared.

Bit 0 of the status field will be set by the system upon completion of the requested function.

Return Codes

The following is a list of codes returned in bits 9-17 of location fdbaddr when the RC or RT option is in effect. Note that error codes 27-33 are now treated as non-fatal.

Return Code	Meaning	Attempted† Function
0	Function successful	A,C,E,P
1	Not used	
2	Local file name already assigned	A
3	Unknown local file name	C,E,P
4	Blank permanent file name	A,C
5	Permanent file directory full	C
6	RBTC full	C,E
7	Permanent file device unavailable	A
10	Latest index not written (for random file)	C,E
11	File is not disk-resident	C
12	Permanent file is not on system	A
13	Cycle referenced does not exist	A
14	Invalid cycle number	A,C
15	Duplicate name/cycle, or 5 cycles already cataloged	C
16	Attempt to catalog non-temporary file	C
17	Not used	
20	Function attempted on non-permanent file	E,P
21	Function attempted on purged file	C,E,P
22	Attempt to catalog an empty file	C
23	Cycle incomplete	A
24	PF is already attached to job	A
25	PF is assigned to another job	A
26	PFD is temporarily full	C
27	No permission to purge this PF	P
30	No permission to add a new cycle	C
31	No correct passwords submitted	A
32	Not used	
33	No extend permission	E

† Key: A = ATTACH, C = CATALOG, E = EXTEND, P = PURGE

Parameters

Bits 0-5 (nn) of each parameter word identify the type of parameter, while the upper 54 bits contain the parameter value. The following table lists the values recognized for nn, the corresponding keywords for the FDB parameter list, and an explanation of the parameter value.

nn (octal)	Parameter Keyword	Explanation of Parameter Value
02	RP	Retention period in days (binary)
03	CY	Cycle number (binary)
04	TK	Turnkey password (display code)
05	CN	Control password (display code)
06	MD	Modify password (display code)
07	EX	Extend password (display code)
10	RD	Read password (display code)
12	SD	ignored
14	ID	Owner identification (display code)
20	PW	Up to five submitted passwords (display code)
21		
22		
23		
24		

Two additional parameters are recognized but cannot be specified with an FDB macro parameter. They must be "hand-coded" into the file definition block.

The "No Duplicate Name" option:

59	5 0
zero	52

If the user attempts to catalog a permanent file name which already exists, the system ordinarily adds a digit prefix to make the name unique. But if the 52 parameter is specified, the file will not be cataloged and error code 15 will be returned to the FDB.

The "Request Status Information" option:

59	5 0
zero	51

When the 51 parameter is specified, status information will be returned when the file is attached. The PP routine PFA will return the following information in the parameter word.

59	53	47	35	5 0
bits	cycle	sd		51

bits   one of the following codes:

58 - cycle incomplete
57 - cycle altered since last dump
56 - cycle dumped in last dump

sd   the subdirectory number

cycle   the cycle number

EXAMPLES

Example 1: Cataloging a file.

```
          IDENT    SAMPLE
          ENTRY    START
TAPE1     FILEB    OBUF,513
OBUF      BSSZ     512
PF        FDB      TAPE1,(MULTI-READ FILE),EX=A,MD=B,CN=C,RP=30
          .
          .
          .
          CATALOG PF
          .
          .
          .
```

Example 2: Specifying the "no duplicate name" option.

```
          IDENT    SAMPLE
          .
          .
          .
PF        FDB      OUTPUT,MYPF,SD=1,TK=SCRAMBLE,RP=999
          .
          .
          .
          SX6      52B
          SA6      PF+1            .STORE 52 PARM IN FDB
          CATALOG PF,RC            .ATTEMPT CATALOG
          SA1      PF              .CHECK RETURN CODE
          AX1      9
          SX5      777B
          BX1      X1*X5
          NZ       X1,PFERROR      .BRANCH IF UNSUCCESSFUL
          .
          .
          .
```

The SD parameter is used as a place-holder, so that the 52 parameter may be inserted into the FDB.

Example 3: Attaching a file.

```
          IDENT    SAMPLE
          ENTRY    START
FDBADDR   FDB      FILE1,(F.B.I. FILE),PW=SUPER,SLEUTH,CY=5
          .
          .
          .
START     ATTACH   FDBADDR
          READ     FILE1,RECALL
          .
          .
          .
```

8.5.6 CONNECT

CONNECT fname,recall

<table>
<tr><td>59</td><td colspan="4">41</td><td>29</td><td>23</td><td>17 0</td></tr>
<tr><td colspan="5"></td><td colspan="2">RJ</td><td>CPC</td></tr>
<tr><td>CON</td><td>1</td><td>r</td><td></td><td colspan="2">0000</td><td></td><td>fname</td></tr>
</table>

The CONNECT function is used to connect a file for terminal input/ouput under MISTIC2. If the program is being run from a batch job, the CONNECT function is ignored. The parameter fname is the location symbol or address of a word containing the display coded file name in L format, with bit 0 clear.

Example:

```
SA2      =5LINPUT
BX6      X2
SA6      TEMP
CONNECT  TEMP,RC        connects file INPUT
```

See Chapter 5 of User's Guide Volume IV for a complete discussion of interactive input/output.

8.5.7 DISCONT

DISCONT fname,recall

<table>
<tr><td>59</td><td colspan="4">41</td><td>29</td><td>23</td><td>17 0</td></tr>
<tr><td colspan="5"></td><td colspan="2">RJ</td><td>CPC</td></tr>
<tr><td>CON</td><td>1</td><td>r</td><td></td><td colspan="2">0001</td><td></td><td>fname</td></tr>
</table>

This function disconnects file fname, where fname is the location symbol or address of a word containing the display coded file name in L format, with bit 0 clear. Files may be connected and disconnected any number of times; redundant CONNECT and DISCONT functions are ignored.

Example:

```
DOUT    DATA    6LOUTPUT
          .
          .
          .
        DISCONT DOUT,RC        disconnects file OUTPUT
```

8.5.8
INTRCOM

INTRCOM loc,recall

59	41			29	23	17 0
				RJ		CPC
RWE	1	r		0000		loc

The INTRCOM function tests whether the program is in batch or interactive mode. The value returned in location loc is 1 if the program is run from batch, and 3 if it is run from MISTIC2.

Example:

```
ITEST   BSSZ    1
          .
          .
          .
        INTRCOM ITEST,RECALL
        SA1     ITEST
        LX1     59-1           test bit 1
        NG      X1,INTJOB      jump if MISTIC2 job
```

8.5.9
CPSTAT

CPSTAT parm,recall

59	41			29	23	17 0
				RJ		CPC
SYS	1	r		0003		parm

CPSTAT returns a table of job information in a 16-word block beginning at location parm. The content and format of this table are given below.

Word	Bits	Contents
1	59-18	Problem number (display code)
	17-1	TTY telephone line number
	0	Complete bit (set on completion of request)
2	59-18	Job sequence number (display code)
	17-0	PN ordinal (identifies the user ID)
3	59-48	Pool ID (display code)
	47-42	Major LSD number in binary (the 31 of 31.03)
	41-36	Minor LSD number in binary (the 03 of 31.03)
	11-0	Last error code
4	59-48	Reprieve flags
	47-24	Reprieve address
	17-12	Tape reservation count
	11-0	Priority
5	59-48	Current RFL value (divided by 100B)
	47-42	Punch copies count
	41-36	Print copies count
6	53-51	Access level
	50-36	CPU time limit
	35-24	Print limit (divided by 10B)
	23-12	Punch limit (divided by 10B)
	11-0	Maximum field length (divided by 100B)
7	23-18	File limit
	17-0	Connect time remaining (seconds)
8	17	PN manager bit (1=yes)
	16	Activity bit (1=PP request is active)
	15-14	LOCK flag *
	13-12	DAYMSG flag *
	11	PROMPT flag (1=ON)
	10	Systems authorization (1=yes)
	9	RERUN flag (1=OFF)
	8-7	AUTORFL flag *
	6-5	Loader flag (1=CP)
	4-3	MAP flag *
	2	REDUCE flag (1=ON)
	1	Tapes reserved (1=yes)
	0	EXPORT job (1=yes)
9	35-12	CPU time (seconds)
	11-0	CPU time (milliseconds)

* Flags are: 0=OFF, 1=ON, 2=PART

Word	Bits	Contents
10	35-12	PPU time (seconds)
	11-0	PPU time (milliseconds)
11	59-36	RA+1 requests
	35-0	PRUs transferred
12	59-0	CM usage, a floating point number representing <64-word blocks> × <milliseconds>.
13	59-0	User ID (MISTIC2 jobs only)

8.5.10 PPTIME

PPTIME loc

59 41 29 23 17 0

<table>
<tr><td colspan="5"></td><td colspan="2">RJ</td><td>CPC</td></tr>
<tr><td>TIM</td><td>1</td><td>1</td><td></td><td colspan="2">0005</td><td></td><td>loc</td></tr>
</table>

This function returns the current elapsed PP time in location loc. Bits 12-35 of loc will contain the number of seconds and bits 0-11 will contain the number of milliseconds.

Example:

```
        MX6     0
        SA6     TIMEHOLD
        PPTIME  TIMEHOLD        store PP time in TIMEHOLD
```

If a time of 12.053 (decimal) were returned, TIMEHOLD would contain the following value.

59 35 11 0

<table>
<tr><td>00000000</td><td>00000014</td><td>0065</td></tr>
</table>

8.5.11 PAUSE

PAUSE parm

59 41 29 23 17 0

<table>
<tr><td colspan="4"></td><td>RJ</td><td>CPC</td></tr>
<tr><td>SMO</td><td>1</td><td>1</td><td colspan="2">0002</td><td>parm</td></tr>
</table>

The PAUSE function swaps out the user's job and displays a two-word (20-character) message to the operator. The job remains swapped out until the operator acknowledges the message by typing GO. Location parm contains the address of the message in bits 30-47.

Example:

```
PARM1    VFD     30/MSG1,30/0
PARM2    VFD     30/MSG2,30/0
MSG1     DIS     2,DROP JOB IF...
MSG2     DIS     2,TAPE LABELS MISMATCH
         .
         .
         .
         PAUSE   PARM1           display first message
         PAUSE   PARM2           display second message
         MSUREQ  TAPE1,RW        request tape
         .
         .
         .
```

8.5.12 DMP

```
         DMP     parm
         DMP     fwa,lwa
```

The DMP function outputs register and memory dumps on file OUTPUT. The DMP macro does not call CPC but generates code which plants a request in RA+1 and loops until its completion. DMP destroys and does not restore the initial contents of A1, X1, A6, and X6. None of the other registers is affected.

The DMP macro call has two forms. If two parameters (fwa and lwa) are specified, the information from location fwa through lwa is dumped onto file OUTPUT. The format of this dump is the same as for a DMP control card. If the first and last word addresses are both zero (e.g., DMP 0,0), the exchange package, RA+0 to RA+100, and the 100B locations before and after the P address are dumped. When only one parameter (parm) is specified, it is assumed to be the address of a word containing the first and last word addresses of the dump in the following format. Bit 0 of this word must be cleared before each DMP request.

59	47	29	11 0
	fwa	lwa	

Example 1: DMP fwa,lwa

```
         DMP     101B,200B       dumps 64 words starting at
                                 location 101 (octal).

         DMP     0,0             dumps all registers plus
                                 the 64 words on either side
                                 of the current P address.
```

Example 2: A DMP subroutine.

```
          IDENT   SAMPLE
            :
          SX1     100B
          SX6     OBUF
          RJ      DUMP              call dump routine to dump from
            :                       location 100B through OBUF
          END     START

          IDENT   DUMP
          ENTRY   DUMP
DUMP      LX1     30                position fwa
          LX6     12                position lwa
          BX6     X1+X6             combine fwa and lwa
          SA6     DMPARM            store in DMP parameter word
          DMP     DMPARM            perform dump
          EQ      DUMP              return
DMPARM    BSSZ    1
          END
```

8.5.13 TAPRES

```
          TAPRES  parm
          TAPRES  parm,val
```

```
59               41           29       23      17                 0
+--------------------------------------+-------------+--------------+
|                                      |      RJ     |      CPC     |
+--------------+--+--+-----+-----------+-----+-------+--------------+
|     TAP      | 1| 1|     |    0000   |     |       parm           |
+--------------+--+--+-----+-----------+-----+----------------------+
```

This function resets the job tape unit reservation in the same manner as the TAPRES control card. The macro has two forms. In both forms parm is the address of a scratch cell. If the second parameter (val) is used to specify the new tape reservation count, its value is stored in bits 12-24 of location parm. Otherwise, it is assumed that location parm already contains the new count. The use of this macro is subject to the rules discussed in Section 6.4.

Examples:

```
TEMP      BSS     1
            :
          TAPRES  TEMP,2            sets tape reservation to 2
            :
          SX6     B7
          LX6     12
          SA6     TEMP
          TAPRES  TEMP              sets tape reservation to
                                    the value in B7
```

8.5.14 RERUN

```
RERUN    parm,val
```

59 41 29 23 17 0

<table>
<tr><td colspan="5"></td><td colspan="2">RJ</td><td>CPC</td></tr>
<tr><td>SYS</td><td>1</td><td>1</td><td></td><td colspan="2">0005</td><td></td><td>parm</td></tr>
</table>

This macro performs the same function as the RERUN control card. The first parameter (parm) is the address of a scratch cell. The second parameter (val) is either ON for RERUN(ON) or OFF for RERUN(OFF). If val is omitted parm should contain the value 2 for ON or 0 for OFF.

Example:

```
RERUN    =STEMP,OFF    disallows rerun
  .
  .
  .
SX6      2
SA6      TEMP
RERUN    TEMP          restores RERUN(ON)
```

8.5.15 EXECM

```
EXECM    parm,RB
```

59 41 29 23 17 0

<table>
<tr><td colspan="5"></td><td colspan="2">RJ</td><td>CPC</td></tr>
<tr><td>IAP</td><td>1</td><td>1</td><td></td><td colspan="2">rb</td><td></td><td>parm</td></tr>
</table>

where: rb = 0001B when RB is specified,
0000B when RB is omitted.

The EXECM macro allows the user to execute a set of SCOPE/HUSTLER control statements contained in a program buffer. EXECM causes the system to read up to 64 words from a specified area of the user's field length and store them in the control card buffer of the control point area. When the program terminates, the system will begin execution of these control statements.

Word parm specifies the address and length of the user's control card buffer as follows:

bit 0	- complete bit
bits 1-11	- word count (length of buffer)
bits 12-29	- first word address of buffer
bit 59	- overflow bit

Unlike the EXEC control card, the EXECM macro allows the user to continue processing the original set of control cards after the alternate set has been executed. When the RB parameter is specified, the control statements supplied by the user's program are inserted between the current program call and the remaining control cards of the job deck.

The system control card buffer is limited to 64 words. If the space remaining after the alternate control cards have been read into this buffer is not sufficient to hold the rest of the original control cards, the overflow bit is set and the RB option is ignored. Since overflow cannot occur when the alternate set merely replaces the original set, the overflow bit need not be tested when RB is omitted.

Example:

```
CCBUF     DATA     C*APLIB,LMT201,*PROG2.*
          DATA     C*COMPASS,I=PROG2.*
          DATA     C*LGO.*
          DATA     C*DISPOSE,TAPE2,PR=A.*
PARM      BSS2     1
          :
          SX6      PARM-CCBUF     X6=word count
          SX7      CCBUF          X7=fwa of buffer
          LX6      1              position word count
          LX7      12             position fwa
          BX6      X6+X7          combine into param word
          SA6      PARM
          EXECM    PARM,RB        inserts control cards from
          :                       CCBUF into the control card
          ENDRUN                  record
```

PROG2 will be retrieved, assembled and executed when this program terminates.

8.5.16 FNTSTAT AND FNTBLOK

The FNTSTAT macro returns information about a specified local file to a pseudo, or real, FET. A pseudo FET can be set up with the FNTBLOK macro as described below. If a real FET is used, the buffer pointers in words 2, 3, 4, and 5 will not be affected by FNTSTAT. This section first describes the format of the two macros and then the format of the information returned.

```
          FNTSTAT blkname,recall
```

blkname the location symbol or address of the table which is to receive the file information. Generally this is the location symbol associated with the FNTBLOK macro call.

recall an optional auto-recall parameter.

FNTSTAT generates the following code:

59	41			29	23	17 0
SA1 blkname					RJ	CPC
SYS	0	r		0031		blkname

blkname FNTBLOK fname,length

blkname the location symbol associated with the first word of the generated block.

fname the name of the local file for which information is to be returned.

length the length of the pseudo FET to be created. The minimum length is 5. If the second parameter is omitted, the length is assumed to be 5.

The FNTBLOK macro expands as follows:

	59	23	17	0
1	fname (left-justified zero-filled)			1
2		length-5		
3	zero			
•				
•				
•				
length	zero			

The information contained in the pseudo FET upon return from FNTSTAT is as follows:

Word	Bits	Contents
1	59-18	File name (left-justified zero-filled)
	17-0	Last code and status
2	59-48	Device type code
	47	Random bit (set if file is random-access)
	35-24	Disposition code
	23-18	Length of block minus five

Word	Bits	Contents
3	59-36	Current PRU position of the file
	35	Set if file does not exist (NO FILE bit)
	34	Set if file is busy (FILE BUSY bit)
	33	Set if file has not been assigned a device
	32	Set if file is empty
	31	Set if file is a permanent file
	30-24	Reserved for future use
	23	Set if CN permission is granted
	22	Set if MD permission is granted
	21	Set if EX permission is granted
	20	Set if RD permission is granted
	19	Set if file is not open for read
	18	Set if file is not open for write
4	59-36	Size of the file in PRUs
5	59-0	Reserved for future use

The user should note the following special conditions:

1) If the file does not exist, i.e., no FNT entry is found, only the NO FILE bit is set. No other fields are changed.

2) If the file is busy, i.e., the complete bit is not set in the FNT, the file is flagged as found, and the FILE BUSY bit is set. All other fields remain unchanged.

3) If bit 33 in word 3 is set, the device type and file size fields are not set.

4) If the file resides on magnetic tape the size field is not set.

5) If the file is a permanent file, the PF permission bits will be set according to the permissions granted, and the PF bit will be set. Otherwise, the PF permission bits have no meaning.

Example:

```
          IDENT     SAMPLE
          ENTRY     START
LOC1      FDB       PF,SORTINPUTFILE,TK=SNOOPY   generates FDB
LOC2      FNTBLOK   PF             generates pseudo FET
          :
START     ATTACH    LOC1           attach SORTINPUTFILE
          FNTSTAT   LOC2,RCL       retrieve status information
          SA1       LOC2+3         fetch file size
          :
```

8.5.17 REPRIEVE AND RBLOCK

The REPRIEVE macro initializes a system utility which allows the user's program to regain control under conditions that would ordinarily cause it to terminate. This function is similar in concept to the OWNCODE capability for file action requests, but it covers a wider range of error conditions.

The format of the macro call is

REPRIEVE addr,p_1,p_2, ... ,p_7

addr . the first word address of the recovery block which is to receive control in the event that one of the selected conditions occurs. A copy of the exchange package and the contents of RA+1 are returned in the first 17 (21B) words of this block; execution begins at addr+21B. The RBLOCK macro may be used to define this block.

p_i specifies the condition(s) under which the program is to be reprieved. The p_i may include any and all of the following.

		octal code
MODE	Arithmetic mode errors	001
CALL	PP call and auto-recall errors	002
TIME	Job limit exceeded	004
DROP	Operator drop or rerun	010
SYSABT	PPU abort	020
CPABT	CPU abort or MISTIC2 user abort	040
NORMAL	Normal termination	100

The REPRIEVE macro generates the following code.

<table>
<tr><td>59</td><td>41</td><td></td><td></td><td></td><td>29</td><td>23</td><td>17   0</td></tr>
<tr><td colspan="5"></td><td colspan="2">RJ</td><td>CPC</td></tr>
<tr><td>RPV</td><td>1</td><td>1</td><td></td><td colspan="2">flags</td><td></td><td>addr</td></tr>
</table>

The 'flags' field contains the sum of the octal codes for the conditions specified by p_1, p_2, etc.

Each successive REPRIEVE call replaces the previous call; they are not stacked, as are the FORTRAN calls to RECOVR. Also, the REPRIEVE call is cleared when the error is recovered, which means that if additional errors are to be recovered, the program must make another call to REPRIEVE.

If a program is reprieved because the time limit, mass storage limit, or job cost limit is exceeded, an additional 5 CPU seconds, 500B PRUs, and/or $1.00 is allowed. But each extension is granted only once.

The RBLOCK Macro

The RBLOCK macro is used to generate the 21B word block in which the RPV routine returns the job exchange package and the contents of RA+1. It also allows the user to specify an area to be checksummed and verified before the "cleanup/recovery" code is executed. If the checksum calculated at recovery time does not match the checksum calculated when the REPRIEVE call was made, the recovery is aborted. The form of the RBLOCK macro is

```
          label    RBLOCK   lwa+1
```

label — the location symbol referenced by the addr parameter of the REPRIEVE call.

lwa+1 — the last-word-address-plus-one of the area to be checksummed. The first word address is label+21B. If this parameter is omitted, the recovery code is not checksummed.

The user code to be executed when an error is recovered begins immediately after the RBLOCK macro. When this code is entered, the 21B word block defined by RBLOCK will contain the following information.

label to label+17B — the exchange package of this job. The format of this block is shown in Section 1.2 of the SCOPE Reference Manual.

label+20B — the contents of RA+1.

The lower 18 bits of location 'label' (register B0) will contain the system error code for the condition that caused the recovery. These codes are listed on the following page.

CAUTION: RBLOCK generates the code:

```
          label    VFD          30/lwa,30/0
                   BSSZ         16
```

Since the location 'label' is overwritten by the exchange package, this word should either be zeroed (for no checksum) or reset to the last word address before requesting a repeated reprieve.

Example:

```
          IDENT     MSG
          ENTRY     START

START     REPRIEVE BLK,CPABT    requests recovery on CPU abort
          ABORT                 causes a CPU abort

BLK       RBLOCK    LWA         generate 17 word block and
                                   request checksum of area
          MESSAGE   LWA,,RCL    print a dayfile message
          ENDRUN
LWA       DATA      C*REPRIEVE WORKS*
          END       START
```

ystem
rror Codes

Octal Code	Error Condition
1	Time limit
2	Arithmetic mode error
3	PPU abort
4	CPU abort
5	PP call error
6	Operator drop
7	Operator kill´
10	Operator rerun
11	Control card error
12	ECS parity error
13	Job Card error
14	Pre-abort
15	Auto-recall error
16	Hung in auto-recall
17	Unauthorized program
20	Insufficient field length
21	File limit exceeded
22	Dollar limit exceeded (1st time)
23	Mass storage limit exceeded
24	User abort (MISTIC2)
25	Input/output error
26	DMP error
27	RTL error
30	Disk parity error
31	WAIT SYSTEM (MISTIC2)
32	Rubouts
33	Dollar limit exceeded (2nd time)
34	Logout

65202g

Appendix A
Character Sets

GRAPHIC	DISPLAY[2] CODE	026 STD CARD CODE	029 STD[1] CARD CODE	EXTERNAL BCD
A	01	12-1	12-1	61
B	02	12-2	12-2	62
C	03	12-3	12-3	63
D	04	12-4	12-4	64
E	05	12-5	12-5	65
F	06	12-6	12-6	66
G	07	12-7	12-7	67
H	10	12-8	12-8	70
I	11	12-9	12-9	71
J	12	11-1	11-1	41
K	13	11-2	11-2	42
L	14	11-3	11-3	43
M	15	11-4	11-4	44
N	16	11-5	11-5	45
O	17	11-6	11-6	46
P	20	11-7	11-7	47
Q	21	11-8	11-8	50
R	22	11-9	11-9	51
S	23	0-2	0-2	22
T	24	0-3	0-3	23
U	25	0-4	0-4	24
V	26	0-5	0-5	25
W	27	0-6	0-6	26
X	30	0-7	0-7	27
Y	31	0-8	0-8	30
Z	32	0-9	0-9	31
0	33	0	0	12
1	34	1	1	01
2	35	2	2	02
3	36	3	3	03
4	37	4	4	04
5	40	5	5	05

GRAPHIC	DISPLAY CODE	026 STD[4] CARD CODE	029 STD[1] CARD CODE	EXTERNAL BCD
6	41	6	6	06
7	42	7	7	07
8	43	8	8	10
9	44	9	9	11
+	45	12	12-8-6	60
-	46	11	11	40
*	47	11-8-4	11-8-4	54
/	50	0-1	0-1	21
(	51	0-8-4	12-8-5	34
)	52	12-8-4	11-8-5	74
$	53	11-8-3	11-8-3	53
=	54	8-3	8-6	13
blank	55	blank	blank	20
,	56	0-8-3	0-8-3	33
.	57	12-8-3	12-8-3	73
#	60	0-8-6	8-3	36
[	61	8-7	12-2-8	17
]	62	0-8-2	11-2-8	32
:	63	8-2	8-2	12[3]
"	64	8-4	8-7	14
—	65	0-8-5	0-8-5	35
!	66	11-0[4]	12-8-7	52
&	67	0-8-7	12	37
'	70	11-8-5	8-5	55
?	71	11-8-6	0-8-7	56
<	72	12-0[4]	12-8-4	72
>	73	11-8-7	0-8-6	57
@	74	8-5	8-4	15
\	75	12-8-5	0-8-2	75
↑	76	12-8-6	11-8-7	76
;	77	12-8-7	11-8-6	77

See Notes on reverse side.

Notes:

[1] In constructing the set of 029 card codes, the ASCII 029 codes were used for all graphics so defined.

[2] Note that there is no character or card code associated with the Display code 00. A Display code 00 and the character following it are sometimes used as a flag by the system. In particular, a Display code 0000 in the lower 12 bits of a central memory word is used as an end-of-line indicator. On a coded output operation to a magnetic tape, a Display code 00 will be converted to external BCD 16, which on input will be converted to Display code 33 (the digit 0). An end-of-line mark, however, will be converted to an external BCD 1632 on output and restored to Display code 0000 on input.

[3] The colon (Display code 63) is not converted to external BCD 00, but to external BCD 12. On input, the external BCD 12 is converted to Display code 33, the digit 0. Thus, the colon is lost on coded 7-track magnetic tape.

[4] The following card code anomalies should be noted.

a. The card punch 11-8-2 is equivalent to 11-0.
b. The card punch 12-8-2 is equivalent to 12-0.
c. The 026 card punch 8-6 and the 029 card punch 0-8-4 are treated as a blank on input.

Translations Between Display Code and ASCII/EBCDIC

DISPLAY CODE		ASCII				EBCDIC			
		UPPER CASE		LOWER CASE		UPPER CASE		LOWER CASE	
OCTAL	CH	CH	HEX	CH	HEX	CH	HEX	CH	HEX
01	A	A	41	a	61	A	C1	a	81
02	B	B	42	b	62	B	C2	b	82
03	C	C	43	c	63	C	C3	c	83
04	D	D	44	d	64	D	C4	d	84
05	E	E	45	e	65	E	C5	e	85
06	F	F	46	f	66	F	C6	f	86
07	G	G	47	g	67	G	C7	g	87
10	H	H	48	h	68	H	C8	h	88
11	I	I	49	i	69	I	C9	i	89
12	J	J	4A	j	6A	J	D1	j	91
13	K	K	4B	k	6B	K	D2	k	92
14	L	L	4C	l	6C	L	D3	l	93
15	M	M	4D	m	6D	M	D4	m	94
16	N	N	4E	n	6E	N	D5	n	95
17	O	O	4F	o	6F	O	D6	o	96
20	P	P	50	p	70	P	D7	p	97
21	Q	Q	51	q	71	Q	D8	q	98
22	R	R	52	r	72	R	D9	r	99
23	S	S	53	s	73	S	E2	s	A2
24	T	T	54	t	74	T	E3	t	A3
25	U	U	55	u	75	U	E4	u	A4
26	V	V	56	v	76	V	E5	v	A5
27	W	W	57	w	77	W	E6	w	A6
30	X	X	58	x	78	X	E7	x	A7

DISPLAY CODE		ASCII				EBCDIC			
		UPPER CASE		LOWER CASE		UPPER CASE		LOWER CASE	
OCTAL	CH	CH	HEX	CH	HEX	CH	HEX	CH	HEX
31	Y	Y	59	y	79	Y	E8	y	A8
32	Z	Z	5A	z	7A	Z	E9	z	A9
33	0	0	30	DLE	10	0	F0	DLE	10
34	1	1	31	DC1	11	1	F1	DC1	11
35	2	2	32	DC2	12	2	F2	DC2	12
36	3	3	33	DC3	13	3	F3	TM	13
37	4	4	34	DC4	14	4	F4	DC4	3C
40	5	5	35	NAK	15	5	F5	NAK	3D
41	6	6	36	SYN	16	6	F6	SYN	32
42	7	7	37	ETB	17	7	F7	ETB	26
43	8	8	38	CAN	18	8	F8	CAN	18
44	9	9	39	EM	19	9	F9	EM	19
45	+	+	2B	VT	0B	+	4E	VT	0B
46	-	-	2D	CR	0D	-	60	CR	0D
47	*	*	2A	LF	0A	*	5C	LF	25
50	/	/	2F	SI	0F	/	61	SI	0F
51	(	(	28	BS	08	(	4D	BS	16
52	)	)	29	HT	09	)	5D	HT	05
53	$	$	24	EOT	04	$	5B	EOT	37
54	=	=	3D	GS	1D	=	7E	IGS	1D
55	SP	SP	20	NUL	00	SP	40	NUL	00
56	,	,	2C	FF	0C	,	6B	FF	0C
57	.	.	2E	SO	0E	.	4B	SO	0E
60	#	#	23	ETX	03	#	7B	ETX	03

DISPLAY CODE		ASCII				EBCDIC			
		UPPER CASE		LOWER CASE		UPPER CASE		LOWER CASE	
OCTAL	CH	CH	HEX	CH	HEX	CH	HEX	CH	HEX
61	[	[	5B	{	7B	[	AD	{	8B
62	]	]	5D	}	7D	]	BD	}	9B
63	:	:	3A	SUB	1A	:	7A	SUB	3F
64	"	"	22	STX	02	"	7F	STX	02
65	—	—	5F	DEL	7F	—	6D	DEL	07
66	!	!	21	SOH	01	!	5A	SOH	01
67	&	&	26	ACK	06	&	50	ACK	2E
70	'	'	27	BEL	07	'	7D	BEL	2F
71	?	?	3F	US	1F	?	6F	IUS	1F
72	<	<	3C	FS	1C	<	4C	IFS	1C
73	>	>	3E	RS	1E	>	6E	IRS	1E
74	@	@	40	`	60	@	7C	`	9A
75	\	\	5C	|	7C	\	BA	|	4F
76	^	^	5E	~	7E	^	AA	~	5F
77	;	;	3B	ESC	1B	;	5E	ESC	27

The shaded translations apply only when reading or writing a 9-track tape in coded mode. These codes, and the characters they represent are not universally agreed upon. The FORM utility may translate these characters using different codes from those listed above.

Notes:

1. The terms "upper case" and "lower case" apply only to the case conversions, and do not necessarily reflect any true "case."

2. When translating from Display code to ASCII/EBCDIC, the "upper case" equivalent character is taken.

3. When translating from ASCII/EBCDIC to Display Code the "upper case" and "lower case" characters fold together to a single Display code equivalent character.

4. All ASCII and EBCDIC codes not listed are translated to Display code 55 (SP).

Punch Card Characters

1. The MSU computer system can read cards prepared on either an IBM 026 or an IBM 029 keypunch. The default mode for any card deck is 026. A 029 deck is recognized when:

   a. the job card contains the number 29 punched in columns 79 and 80, which indicates that the job card and all following cards are to be read in 029 mode; or

   b. an end-of-section card (a 7/8/9 card) contains the number 29 punched in columns 79 and 80, which indicates that all subsequent non-binary cards are to be read in 029 mode.

   In either case, (a) or (b), when an end-of-section card containing the number 26 punched in columns 79 and 80 is encountered, the card reader will return to 026 mode.

2. Illegal Hollerith card punch combinations are converted to legal codes so that they can be processed This conversion may or may not yield the intended character. **There is no indication of error.**

3. Three types of error diagnostics can occur when cards are read, all of which are headed by:

   xxxxxxRC.yyyyyyCARD: message

   xxxxxx the section in which the error was encountered. The sections are indicated by decimal numbers, starting with the control section as section 0.

   yyyyyy the number (decimal) of the card within the indicated section.

   message one of the following messages:

   MODE CHANGE means that BCD and binary cards are intermixed within a single section.

   CHECKSUM ERROR indicates a mispunched binary card. The deck should be discarded and regenerated.

   SEQUENCE ERROR indicates a card is out of sequence in a binary deck. If there is more than one card with the same sequence number, remove the first one; otherwise repunch the binary deck.

Appendix D
Software Support

Name	Title	Support
APEX	Linear Programming System	II
APL	APL Language Processor	II
APLIB	User Library Maintenance	III
AUTHORF	Authorization File Maintenance	I
BASIC	BASIC Language Processor	I
8-Bit Subroutines	IBM-CDC Conversion Routine	I
CALCOMP	CALCOMP Plotting Library	II
COBOL	COBOL Compiler	I
COMPASS	6500 Assembly Language	I
CRM	Cyber Record Manager	I
DMS-170	Database Management System	I
EDITOR	Interactive Text EDITOR	I
EISPACK	Eigen Analysis Package	III
FORM	File Organization Record Manager	I
FTN	FORTRAN Extended Compiler	I
GASP	Simulation Package	III
GCS	Graphics Compatibility System	II
HAL	HUSTLER Auxiliary Library	I
IMSL	International Math/Stat Library	I
KWIC	Key Word In Context	III
LIBEDIT	Loader Library Maintenance	I
LISP	List Processing Language	III
MIMIC	Ordinary Differential Equation Solver	III
MNF	MiNnesota FORTRAN Compiler	I
NETFLOW	Network Flow Analyzer	III
PASCAL	PASCAL Language Compiler	II
PERTTIME	PERT Analyzer	III
SCOPE/HUSTLER	MSU 6500 Batch and Interactive Oper. System	I
SIMULA	Simulation/Algorithmic Language	III
SNOBOL	String Manipulation Language	III
SORTMRG	Sorting/Merging Package	I
SPSS	Statistical Package for the Social Sciences	I
SPURT	Simulation Package	III
STAT	Statistical Analysis System	II
TEKTRONIX	Graphics Software Package	II
UPDATE	Library Preparation and Maintenance	I

Appendix E
DISPOSE Output Source Codes

Each job in the 6500 system has associated with it a 7-character sequence number. The first character of the job sequence number designates the source of entry into the system for that job. After the job begins execution, the second character of the sequence number indicates the destination for the job output. The destination is normally selected automatically and depends on the source of entry.

The DISPOSE utility allows a user to direct output to various printers and to request special handling of output. This is accomplished by requesting one of the output source codes listed below on the DISPOSE control statement. The letter requested will become the second character of the job sequence number.

Note: In order for a user to dispose output to any of these sources he/she must have been previously authorized for access to the given source.

A High-speed printer located in Room 208 Computer Center.

B Jobs to be printed at the central site (207-208 Computer Center).

E,F,G Output that requires special attention. Any jobs routed to E, F or G will not be printed until the user personally requests it through the Service Window in Room 208.

H Attended queue; output directed to the H source will not be filed in the self-service

K CPS plot assignments. Plot output from the 10 queue will be attached to print output before it is filed in the self-service print bins in Room 208.

L Output to be printed on 8.5 inch wide by 11 inch long unlined paper with perforated feed holes.

P Purge queue. This allows users to eliminate the cost of printing files they do not want. Any print files with a P destination will not be printed. These jobs will be purged from the system after execution.

Q Quality print. Output will be printed on standard stock using the best available printer with particular care given to forms alignment.

U Physics remote batch terminal.

V Engineering remote batch terminal.

W Physical Plant remote batch terminal.

X Chemistry remote batch terminal.

Y High-speed remote terminal in Room 208.

Z Public dial-up remote batch port; this source is assigned to many users and caution is advised when using it. Contact the Manager of Operations (355-5046) for details.

Appendix F
Printers and Carriage Controls

Printers and Character Sets

The MSU computer system has two types of printers available: 96-character and 63-character. Both printers use ASCII graphics, either full ASCII or the CDC 63-character ASCII Subset. The 96-character printer has both upper and lower case alphabetic characters, digits and special characters, including vertical bar, accent grave, braces and tilde. The 63-character printer has upper case alphabetic characters, digits and a smaller set of special characters. Both printers will accept ASCII and Display Code data.

If full ASCII data is sent to the 63-character printer; lower case alphabetic characters will print as upper case, accent grave (`) as commercial at (@), braces ({}) as brackets ([]), vertical bar (|) as backslash (\), and tilde (~) as caret (^).

Carriage Controls

The output character set is determined on a line-by-line basis. The first four bits of each line are examined for a carriage control. If these are all zero, the code is ASCII; otherwise it is Display Code, and the first character of the line is processed as a carriage control.

A carriage control is not printed unless it is not recognized as a carriage control character. The carriage control specifies the spacing between consecutive lines. Column 1 of the output will contain the second character of the internal print line.

Central Site Carriage Controls

The following table lists the action specified by each carriage control character for the central site printers. A pre-print character causes the action to be taken before the line is printed. A post-print character causes a line feed, followed by the printing of the line, followed by the specified action. In this table, form refers to physical paper while page refers to the normal print area. (LPI stands for lines per inch.)

Pre-print	Action	Lines	Post-p
Character		Printed	Chara
1	Eject to top of next page	4	H
2	Skip to next 1/2 page	4,34	I
3	Skip to next 1/3 page	4,24,34 (6LPI) or 5,32,59 (8LPI)	J
4	Skip to next 1/4 page	4,19,34,49	D
5	Skip to top of form	1	E
6	Skip to next 1/6 page	4,14,24,34,44,54 (6LPI) or 5,18,32,45,59,82 (8LPI)	F
7	Skip to last line of page		G
8	Space 1 and set auto-page eject		R
9	Space 1 and clear auto-page eject		Q
0	Double space		
-	Triple space		(
+	No space (overprint)		
S	Page eject and select 6 lines/inch		
T	Page eject and select 8 lines/inch		
others	Single space		

Notes:

1. Any pre-print skip operation of 1, 2, or 3 lines that follows a post-print action will be reduced to a skip of 0, 1, or 2 lines.

2. When auto-page eject (the default mode) is in effect, there is a maximum of 61 lines per page in 6 lines/inch, or 81 lines per page in 8 lines/inch. Assuming the paper is properly aligned, the top-of-page will correspond to line 4 of the form, and the printer will eject to the next page after printing on line 64 of the form. If you skip to top-of-form (line 1), the auto-page eject is cancelled for that page.

3. The 1/2, 1/3, 1/4, and 1/6 page positions are based on 60 lines per page, starting at line 4 of the form.

Remote Batch Carriage Controls

The carriage control characters recognized by the remote batch terminal printers appear in the following list. All remote batch carriage controls act before the line is printed.

Carriage Control	Action
1	Eject to top of next page
0	Double space
+	No space (overprint)
all others	Single space

Interactive Carriage Controls

As with lines sent to a printer, the first character of each line of ASCII Fancy (AF) or Display Code (OM) sent to the terminal acts as a carriage control and is not printed. Files coded in ASCII (AS) or Binary (BI) contain no carriage controls. All interactive carriage controls act before the line is printed.

Carriage Control	Action
1	Triple Space
0	Double space
+	No space (overprint)
-	Triple space
,	No action (i.e. no carriage return)
all others	Single space

Appendix I
Magnetic Tape Labels

The standard SCOPE tape labels described in this appendix were designed to conform to the proposed USA Standard for Magnetic Tape Labels and File Structure for Information Interchange submitted by the X.3.2/457 Committee on November 28, 1966. (Note: the format of the ANSI standard labels has been extended and slightly modified since this date.)

The only local modification to the SCOPE label format is the inclusion of a problem number field in the volume header label.

SCOPE labels are recorded at the same density as data on the tape. Each label consists of an 80-character block written in external BCD codes on 7-track tapes or ASCII codes on 9-track tapes.

SCOPE Labels

The layout of labels, data, and file gaps for SCOPE tapes is shown below. The four-character label identifier is used to represent a label block, and the asterisk is used to represent a file gap.

Single-Volume File:

VOL1	HDR1	*	... Data Blocks ...	*	EOF1	*	*	

Multi-Volume File:

VOL1	HDR1	*	... First Volume Data ...	*	EOV1	*	*		reel 1
VOL1	HDR1	*	.. Second Volume Data ..	*	EOF1	*	*		reel 2

Multi-File Volume[1]:

VOL1	HDR1	*	... File A ...	*	EOF1	*	HDR1	*	... File B ...	*	EOF1	*	*	

Multi-Volume Multi-File[1]:

VOL1	HDR1	*	... File A ...	*	EOF1	*	HDR1	*	... File B ...	*	EOV1	*	*		reel 1

VOL1	HDR1	*	... Continuation of File B ...	*	EOV1	*	*		reel 2

VOL1	HDR1	*	... Last of File B ...	*	EOF1	*	HDR1	*	... File C ...	*	EOF1	*	*		reel 3

[1] Multifile tapes are not supported at MSU and their use is discouraged.

SCOPE Volume Header Label

Character	Content[1]	Description	Default
1-3	VOL	Label identifier for volume header label	VOL
4	1	Label number	1
5-10	nnnnnn	Visual reel name; 6 characters giving the identifier printed on the tape reel.	specified by the REQUEST statement
11	a	Security; not processed by SCOPE/HUSTLER	blank
12	a	Volume density: 7-track: Blank or 0 556 cpi 1 200 cpi 2 800 cpi 9-track: blank 1600 cpi 2 800 cpi	blank
13-19	nnnnnnn	Problem Number; 6 or 7 digits specifying the problem number needed for write permission on the tape. If zero, any problem number is authorized to write.	taken from columns 7-13 of the job PNC
20-79	blank	Reserved for future use	blank
80	1	Label standard level	1

[1] An 'n' represents a numeric character (0 through 9). An 'a' represents any of the alphabetic, numeric, or special characters listed as Display code characters in Appendix A.

SCOPE File Header Label

Character	Content	Description	Default
1-3	HDR	Label identifier for file header label	HDR
4	1	Label number	1
5-21	17(a)	File label name; up to 17 characters identifying this file	blank
22-27	aaaaaa	Multifile set name; up to six characters starting with a letter. Multifile tape set processing is not available under SCOPE/HUSTLER.	blank
28-31	nnnn	Reel number; 0001 to 9999, incremented by 1 after each volume trailer label written	0001 for first reel and incremented by 1 for each subsequent reel.
32-35	nnnn	Multifile position number; gives position of a file within a multifile set. Not used by SCOPE/HUSTLER.	0001
36-39	blank	Reserved for future use	blank
40-41	nn	Edition number; 00 to 99, distinguishing successive generations of files with same file label name.	01
42	blank	Reserved for future use	blank
43-47	yyddd	Creation date: yy year ddd Julian date (001 to 366)	current date
48	blank	Reserved for future use	blank
49-53	yyddd	Expiration date; file can be overwritten without a warning message when this date equals the current date. Same format as creation date.	current date
54	a	Security; not processed by SCOPE/HUSTLER	blank
55-60	000000	Block count; must be 000000	000000
61-80	blank	Reserved for future use	blank

SCOPE File Trailer Label

Character	Content	Description	Default
1-3	EOF	Label identifier for file trailer label. This label is a SCOPE end-of-information indication.	EOF
4	1	Label number	1
5-54	optional	Identical to corresponding characters of the preceding file header label	
55-60	nnnnnn blanks	Block count; the number of data blocks (physical records of data) written since the preceding HDR label group.	
61-80		Reserved for future use.	blanks

Volume Trailer Label

Character	Content	Description	Default
1-3	EOV	Label identifier for volume trailer label; signals end-of-reel	EOV
4	1	Label number	1
5-54	optional	Identical to corresponding characters of the preceding file header label	
55-60	nnnnnn	Block count; the number of data blocks (physical records of data) written since the preceding HDR label group.	
61-80	blanks	Reserved for future use	blanks

3600 Labels (Obsolete)

Although obsolete, 3600 labels may be read by SCOPE/HUSTLER.

The layout of labels, data, and tape marks for 3600 labeled tapes is shown below. The following symbols are used:

HDR	header label
EOF	end of file label
EOT	end of tape label
EOS	end of system label
*	file gap

Single-Volume File:

HDR	... File A ...	*	EOF	*	*	EOS	*	

Multi-File Volume:

HDR	... File A ...	*	EOF	*	HDR	... File B ...	*	EOF	*	*	EOS	*	

Multi-Volume File:

HDR	... File A ...	*	EOT	*	*		reel 1

HDR	.. Continuation of File A ..	*	EOF	*	*	EOS	*		reel 2

3600 Header Label

Character	Content	Description	Default
1	n	Density: 2 200 bpi 5 556 bpi 8 800 bpi	5
2-3	()	Label identifier for header label	()
4-5	nn	Logical unit number; blank if created by SCOPE/HUSTLER	blank
6-8	nnn	Retention cycle; number of days file is to be protected against over-writing	000
9-22	14(a)	File label name	blank
23-24	nn	Reel number; 00 to 99, incremented by one after each EOT label written	01
25-30	mmddyy	Creation date: mm month dd day yy year	current date
31-32	nn	Edition number; 01 to 99, identifying successive generations of files with the same file label name.	01
33-40	8(n)	Visual reel name; specifies the identifier printed on the tape reel	specified by the REQUEST statement
41-48	8(n)	Problem number; 6 or 7 digits specifying the problem number authorized to write on the tape. If zero, any PN is authorized for writing.	taken from columns 7-13 of the job PNC
49-80	32(a)	User-supplied information	blanks

3600 Trailer Labels

Character	Content	Description	Default
1-2	EO	First two characters of label identifier	EO
3	F,T, or S	Specifies trailer type; F end of file label used by SCOPE as an end-of-information mark T end of tape label used when file is continued onto another reel S end of system label written after the last EOF label on a volume	F,T, or S
4-8	nnnnn	Block count; the number of physical records written since the beginning of the reel.	
9-80	72(a)	User-supplied information	blanks

Appendix J
Control Statement Summary

The following summary lists in alphabetical order all control statements on the SCOPE/HUSTLER operating system (i.e. not on the HUSTLER Auxiliary Library), except those that are unique to the interactive system (see the *Interactive System User's Guide*, Appendix G). Accompanying the format description of each control statement is a brief explanation of its function and a reference to the user's guide (UG) or reference manual (RM) that contains a complete discussion of its use.

Notation

The following notation is used in this summary:

<table>
<tr><td>UPPER CASE</td><td>must appear as shown</td></tr>
<tr><td>lower case</td><td>must be supplied by user</td></tr>
<tr><td>|</td><td>separates alternate forms (logical OR)</td></tr>
<tr><td>{}</td><td>encloses alternate forms</td></tr>
<tr><td>[]</td><td>encloses optional forms</td></tr>
<tr><td>[,...]</td><td>encloses order-independent optional forms. Item may be repeated, separated by commas.</td></tr>
<tr><td>,[ ]</td><td>encloses order-dependent optional forms (i.e. parameters must appear in the order shown. If a parameter is specified, all preceding commas must be included even if their corresponding parameters are not specified.)</td></tr>
<tr><td>[=...]</td><td>optional item may be repeated, separated by equal signs.</td></tr>
<tr><td>[,param]</td><td>delimiter is used only when param is specified</td></tr>
<tr><td>,[param]</td><td>delimiter is used even when param is omitted (i.e. the parameter is order-dependent)</td></tr>
<tr><td>―――</td><td>underscores default form</td></tr>
<tr><td>▬▬▬</td><td>underscores abbreviation</td></tr>
</table>

```
id[,CMwords][,Tseconds][,Lpages][,Ccards][,JCcents]
   [,RGrg][,MSprus][,MTtapes][,NTtapes][,PNpn]
   [,M64][,INIT|,NOINIT].
```

The job card specifies the user ID and job limits.

SCOPE/HUSTLER RM
Section 3.2.3

```
PW=password
```

The password card, which follows the job card, provides a measure of security for the user's problem number account.

SCOPE/HUSTLER RM
Section 3.2.4

```
name[,exparam[,...]].
```

Loads and executes program from local file 'name' or from a user library.

Cyber Loader RM (CDC No. 60429800)
Section 2
SCOPE/HUSTLER RM
Section 7.9.5

```
APL.
```

Calls the APL interpreter.

SCOPE/HUSTLER RM
Section 7.16.1

```
APLIB,[L|U]{MT|TT|PF}libname[,{I|N|T|U|Z}*[lfn]]
        [,L*[ent]][,[[A|C|G|P]*]ent]
        [,...][,[A|C|G|P]R*ent=lfn][,...].
```

Creates and maintains a user library as a permanent file, magnetic tape, or both; also retrieves specified library files onto user-defined local files.

SCOPE/HUSTLER RM
Section 7.15.1

```
ATTACH,lfn,pfn[,CY=cy][,MR=n][,PW=pw[,...]].
```

Retrieves permanent file pfn as local file lfn and specifies passwords needed to gain desired types of access permissions.

SCOPE/HUSTLER RM
Section 5.2.2, 7.14.1

```
AUTHORF[,afdirective|,optional parameters].
```

Allows PN managers to manipulate and display the contents of the Authorization File, and allows users to change their passwords and display information about their accounts.

SCOPE/HUSTLER RM
Section 2.5, 7.2.1
Interactive System UG
Section 2.8 (interactive use)

```
AUTORFL[,ON|,OFF|,PART].
```

Determines the field length at which size-sensitive system routines are executed. 'AUTORFL,OFF.' should be used with extreme caution.

SCOPE/HUSTLER RM
Section 7.11.1

```
BASIC[,AS[=0]][,B[=binlfn|0]][,BL][,DB={0|B|DL|TR}]
        [,E[=errlfn]][,EL={F|W}][,GO[=0]][,I[=inlfn]]
        [,J=[inputlfn|0]][,K=[printlfn|0][,listlfn|0]
        [,LO=[S|O|0][/...][,PD=[6|8]][,PS=n].
```

Calls the BASIC version 3 compiler.

BASIC RM (CDC No. 19980300)
SCOPE/HUSTLER RM
Section 7.16.2

BKSP,lfn[,n].

SCOPE/HUSTLER RM
Section 7.7.1

Backspaces a file a specified number of sections.

CATALOG,lfn,pfn[,BC=n][,CN=cnpw][,CY=cy]
[,EX=expw][,ID=id][,MD=mdpw][,MR=n]
[,PW=cnpw[,tkpw]][,RD=rdpw][,RP=n][,TK=tkpw].

SCOPE/HUSTLER RM
Section 5.2.1, 7.14.2

Catalogs local file lfn as permanent file pfn and specifies passwords that will be needed to gain various types of access permissions on subsequent ATTACH requests.

COBOL[,A][,[B={binlfn|0]][,BUF][,C][,D][,DB][,DB1]
[,E=prog][,F][,H][,I[=srclfn]][,K={sslfn|0}]
[,L[op][={listlfn|0}]][,N][,OB=ovllfn]
[,S[=srclfn]][,SUB][,SUBM][,T][,U][,V][,W][,Z].

COBOL RM
SCOPE/HUSTLER RM
Section 7.16.3

Calls the COBOL version 4 compiler.

COMBINE,inlfn,outlfn,n.

SCOPE/HUSTLER RM
Section 7.5.1

Reads n sections from file inlfn and writes them as one section (level 0) on file outlfn.

COMMENT,user comments.

SCOPE/HUSTLER RM
Section 7.12.2

Prints user comments in the job dayfile.

COMPARE,lfn$_1$,lfn$_2$,[nsec],[lev]
,[wdprec],[maxerr].

SCOPE/HUSTLER RM
Section 7.6.1

Determines whether the contents of lfn$_1$ and lfn$_2$ are identical.

COMPASS[,A][,B={binlfn|0}][,D][,F=val]
[,G={syslfn|syslfn/ovl|0}][,I=[assmlfn]]
[,L={listlfn|0}][,L{6|8}][,LO={opt|$$$$|0}]
[,ML[=modlev]][,N][,NN][,O=[listlfn|0]][,P]
[,PC[=strng]][,S[={ovl|lib/ovl|0}]][,X[=extlfn]]
[,Y[=WARN]].

COMPASS v3 RM
(CDC No. 60492600)
SCOPE/HUSTLER RM
Section 7.16.4

Calls the COMPASS version 3 assembler.

COPIES,{OUTPUT|PUNCH},[n].

SCOPE/HUSTLER RM
Section 7.12.3

Requests n additional copies of either file OUTPUT or the local punch files.

COPY,[inlfn],[outlfn].

SCOPE/HUSTLER RM
Section 7.5.2

Copies file inlfn onto file outlfn from current position to end-of-information or double end-of-partition. Both files are then backspaced over the last EOP.

COPY{BR|CR|BF|CF},[inlfn],[outlfn],[n].

SCOPE/HUSTLER RM
Section 7.5.4

COPYBR copies n binary sections from file inlfn to file outlfn.
COPYCR copies n coded sections from file inlfn to file outlfn.
COPYBF copies n binary partitions from file inlfn to file outlfn.
COPYCF copies n coded partitions from file inlfn to file outlfn.

COPYBCD,[inlfn],[outlfn],[n].

SCOPE/HUSTLER RM
Section 7.5.3

Copies coded file inlfn to a coded magnetic tape file outlfn, writing each unit record of file inlfn as a discrete block of file outlfn.

COPYCL[,COMPILE=[complfn|0]][,INPUT[=inlfn]]
[LIST=[listlfn|0]][,NPL=[npllfn|0]]
[,OPL=[oplfn|0]].

COBOL RM
Section II-12
SCOPE/HUSTLER RM
Section 7.15.3

Creates, maintains, and updates COBOL source libraries for use with the COPY and INCLUDE statements of the COBOL language.

COPYL,[oldlfn],[modlfn],[newlfn],[last],[R|A|T|E].

SCOPE/HUSTLER RM
Section 7.5.5

Replaces selected routines from file oldlfn with routines from file modlfn and writes the updated file to the file newlfn. This statement is used to update files containing several binary relocatable sections.

COPYLM,[oldlfn],[modlfn],[newlfn],[last],[R|A|T|E].

SCOPE/HUSTLER RM
Section 7.5.5

Replaces selected routines from file oldlfn with routines from file modlfn and writes the updated file to the file newlfn (allowing multiple replacement). This statement is used to update files containing several binary relocatable sections.

COPYN,fmt,outlfn,inlfn[,...].

SCOPE/HUSTLER RM
Section 7.5.6

Copies sections from up to 10 input files onto file outlfn. Directives contained in the corresponding data section determine the final composition of file outlfn.

COPYS{AF|AP|AR|AS|BF|BR|BS|CF|CP|CR|CS},[inlfn]
,[outlfn][,n].

SCOPE/HUSTLER RM
Section 7.5.7

Formats a file that is to be listed on a line printer.

COPY8P,inlfn,outlfn[,BLKSIZE=n][,CODE=[A|C]]
[,FMT=opt][,FOLD][,LRECL=n][,RECFM=rf].

8-Bit Subroutines RM (CDC Nc
60495500)
Chapter 5
SCOPE/HUSTLER RM
Section 7.5.8

Copies IBM 360/370 print files to Control Data-compatible print files.

DAYFILE[,F].

Saves on file DAYF all dayfile messages accumulated since the previous DAYFILE statement of this form or start of job, prints up to 11 most recent lines.

SCOPE/HUSTLER RM
Section 7.13.1

DAYFILE,[fromline,toline[,F]].

Reads contents of file DAYF from previous DAYFILE call; prints messages issued prior to the last 11 lines.

SCOPE/HUSTLER RM
Section 7.13.1

DAYMSG,{ON|OFF|PART}.

Determines what type of messages will be displayed in the dayfile (batch) or at the terminal (interactive).

SCOPE/HUSTLER RM
Section 7.12.4

DFRCV,L=loglfn[,A=areaid][,C=ckptno]
[,D=startdate[/enddate]][,H=ept/libname]
[,O=outlfn][,U=userid].

A data base utility to recover a data base.

Data Base Utilities RM (CDC N
60498800)
Chapter 4
SCOPE/HUSTLER RM
Section 7.19.2

DFRST,L=loglfn[,A=areaid][,C=ckptno]
[,D=startdate[/enddate][,H=ept/libname]
[,O=outlfn][,U=userid].

A data base utility to restore a data base.

Data Base Utilities RM (CDC N
60498800)
Chapter 4
SCOPE/HUSTLER RM
Section 7.19.2

DISPOSE,lfn,dis[=dest][,C=cpy][,L=lmt][,I=acctlfn].

Releases local file lfn and places it in the print, punch, or input queue. User can specify output site (dest), copies count, and page or card limit, in addition to disposition (dis).

SCOPE/HUSTLER RM
Section 7.4.1
Interactive System UG
Chapter 7 (interactive use)

DISPOSE,*lfn,dis.

Assigns disposition to file lfn, but does not release the file until end-of-job.

SCOPE/HUSTLER RM
Section 7.4.1
Interactive System UG
Chapter 7 (interactive use)

DISPOSE,**,dest.

Directs job output to specified site (does not release the file).

SCOPE/HUSTLER RM
Section 7.4.1

DMP[[,fwadump],lwadump].

'DMP.' produces a standard dump. The options fwadump and lwadump specify limits on the user's field length to be dumped.

SCOPE/HUSTLER RM
Section 7.13.2

DMPX,{ON|OFF}.

'DMPX,ON.' selects automatic dump option for abnormal termination. Default is ON for central site jobs, and OFF for remote jobs.

SCOPE/HUSTLER RM
Section 7.13.3

EDITOR[,I=inlfn][,L=listlfn][,E=ewflfn]. *Calls text editor (from batch) to process directives and text lines on file inlfn using the work file ewflfn.*	Interactive System UG Chapter 3 (complete description) SCOPE/HUSTLER RM Section 7.8.1 (batch use)
ERRS[,ALL][,F][,I=inlfn][,NA][,NI][,NS][,O=outlfn][,PG=n][,S]. *Scans a file for program listings and generates an error summary for those that contain error diagnostic messages.*	SCOPE/HUSTLER RM Section 7.13.4
ESTMATE[,KS=ksize][,MI=minrecsz][,MR=maxrecsz][,NR=nrec]. *Aids in calculating indexed block size, data block size, and buffer size for indexed-sequential files.*	Cyber Record Manager (CDC No. 60499300) Advanced Access Methods RM SCOPE/HUSTLER RM Section 7.18.1
EXEC,execlfn. *Initiates execution of control statements from the first section of file execlfn. Remaining control statements on INPUT are ignored.*	SCOPE/HUSTLER RM Section 7.1.2
EXECUTE,[ept][,exparam[,...]]. *Completes loading and linking of programs in memory, and then initiates execution starting at entry point ept.*	Cyber Loader RM (CDC Nc 60429800) Section 2 SCOPE/HUSTLER RM Section 7.9.1
EXIT[,S][,C[,U]. *Precedes a group of control statements to be executed in the event of a fatal job error. 'EXIT,S.' must be used if the user wishes to continue processing after control statement errors or loader aborts caused by compilation/assembly errors.*	SCOPE/HUSTLER RM Section 7.1.3
EXTEND,lfn. *Makes permanent an extension written to an attached permanent file.*	SCOPE/HUSTLER RM Section 5.2.4, 7.14.3
FILE,[lfn][,descriptors]. *Supplies File Information Table values after a source language program is compiled or assembled and before the program is executed.*	Cyber Record Manager R (CDC No. 60307300) Section 2 SCOPE/HUSTLER RM Section 7.4.2
FILES[,O=outlfn]. *Lists the names of all files assigned to the job.*	SCOPE/HUSTLER RM Section 7.4.3 Interactive System UG Section 2.7.2 (interactive use)
FORM[,I=dirlfn][,OWN=binlfn][,L=listlfn]. FORM[,INP=inlfn][,OUT=outlfn][,L=listlfn]. *A file management utility callable by control statements, FORTRAN, COMPASS, or COBOL, which performs a variety of data manipulations.*	FORM RM (CDC No. 6049620 SCOPE/HUSTLER RM Section 7.5.10

```
FTN[,A[=0]][,B=[binlfn|0]][,BL[=0]][,C[=0]]
        [,D[={dbglfn|0}]][,E[={editlfn|0}]][,EL=errlev]
        [,ER[=0]][,G[={syslfn|syslfn/ovl|0}]][GO[=0]]
        [,I[=inlfn]][,L=[listlfn|0]][,ML=nnn]
        [,OL[=0]][,OPT=n][,P[=0]][,PD={6|8}]
        [,PL=n][,PS=nn][,PW=nn][,Q[=0]]
        [,R=n][,ROUND[={opt|0}][,S[={ovl|lib/ovl|0}]]
        [,SEQ[=0]][,SL[=0]][,SYSEDIT[=0]]
        [,T[=0]][,TS][,UO[=0]][,X[=extlfn]]
        [,Z[=0]].
```

Calls the FORTRAN Extended version 4 compiler.

FORTRAN v4 RM
(CDC No. 60497800)
SCOPE/HUSTLER RM
Section 7.16.5

```
FTN5[,ANSI[=0|={F|T}][,ARG[=0|=opt]][,B[=0|=binlfn]
        [,BL[=0]][,CS[=FIXED|=USER]]
        [,DB[=0|={ER|ID|PMD|SB|SL|ST|TB}]]
        [,DO[=0|={LONG|OT}]][,DS[=0]][,E[=errlfn]]
        [,EL[={T|W|F|C}]][,ET[=0|={T|W|F|C}]]
        [,G[=0|={syslfn|syslfn-secname}]][,GO[=0]]
        [,I[=inlfn]][,L[=listlfn]][,LCM[={D|G|I}]]
        [,LO[=0|=opt[/opt]]][,ML[=0|=str]]
        [,OPT[=0|=1|=2|=3]][,PD[={6|8}]][,PL[=n]]
        [,PN[=0]][,PS=n][,PW[=n]][,QC[=0]]
        [,REW[=0|=opt[/opt]]][,ROUND[=0|=opt[/opt]]]
        [,S[=0|=X]][,SEQ[=0]][,X[=xlfn]].
```

Calls the FORTRAN Extended version 5 compiler.

FORTRAN v5 RM
(CDC No. 60481300)
SCOPE/HUSTLER RM
Section 7.16.6

```
F45[,CC[=*|=C]][,CI[=0|=idname]][,DD[=0]][,ET[=0]]
        [,I[=inlfn]][,L[=0|=listlfn]][,LO[=S|=F|=E|=M]]
        [,MC[=$char$]][,MD][,P[=0|=srclfn]][,PD[=6|=8]]
        [,PO[=S|=F|=M]][,SI[=0]][,SO[=0|=n1/n2/n3]].
```

Converts valid FORTRAN 4 source statements to FORTRAN 5 source statements

FORTRAN Extended v4
to FORTRAN v5
CONVERSION AID PROGRAM
RM
(CDC No. 60483000)
SCOPE/HUSTLER RM
Section 7.16.7

HAL.

Satisfies external references on file LGO using subprograms from the HUSTLER Auxiliary Library.

HAL RM
Section 2.1
SCOPE/HUSTLER RM
Section 7.15.4

HAL,[*]lfn=sub[,...].

Copies binary relocatable subprograms from the HUSTLER Auxiliary Library onto file lfn.

HAL RM
Section 2.1
SCOPE/HUSTLER RM
Section 7.15.4

HAL,pname[,param list].

Loads and executes a main program overlay from the HUSTLER Auxiliary Library.

HAL RM
Section 2.2
SCOPE/HUSTLER RM
Section 7.15.4

HAL,*pname[,...].

Copies one or more main program overlays from the HUSTLER Auxiliary Library to local files.

HAL RM
Section 2.2
SCOPE/HUSTLER RM
Section 7.15.4

HAL,[*]dname[,...].

Copies one or more data files from the HUSTLER Auxiliary Library onto local files.

HAL RM
Section 2.3
SCOPE/HUSTLER RM
Section 7.15.4

HAL,[*]lfn=dname[,...].

Copies data file(s) from the HUSTLER Auxiliary Library onto file lfn.

HAL RM
Section 2.3
SCOPE/HUSTLER RM
Section 7.15.4

HAL,E*dname.

Retrieves a data file from the HUSTLER Auxiliary Library and performs an EXEC on that file, i.e., 'EXEC,dname.'

HAL RM
Section 2.3
SCOPE/HUSTLER RM
Section 7.15.4

HAL,L*userlib[,PW=rdpw][,params].

Retrieves subprograms, main programs, and data files from a user auxiliary library. All of the preceding HAL parameter forms are valid for params.

HAL RM
Chapter 2

HELP[,CC|NOCC][,CY=nn][,O={listlfn|0}],
[,OC=[AF|DC]][,PD=[6|8]][,S=[srclfn|0]]
[[moddate=][cat=][ND*][prfx*][keywrd]].

HELP is used to obtain 1) descriptions of programs and data files in the HUSTLER Auxiliary Library, 2) descriptions of SCOPE/HUSTLER control statements, interactive commands, and EDITOR directives, 3) Computer Laboratory announcements of problems, policies, and schedules.

HAL RM
Section 3.1
SCOPE/HUSTLER RM
Section 7.17.1

HELP,L*usrlib[,PW=rdpw][,params].

Obtains documentation for user auxiliary libraries. All of the preceding HELP parameter forms are valid for params.

HAL RM
Section 3.1

Individual programs on the HUSTLER Auxiliary Library are documented either in Chapter 7 or in the HELP file (see Section 7.17.1).

IXGEN,prilfn[,dirlfn].

Creates an index file for alternate key access of an existing indexed-sequential (IS), direct-access (DA), or actual-key (AK) file.

Cyber Record Manager (CD
No. 60499300)
Advanced Access Methods RM
SCOPE/HUSTLER RM
Section 7.18.2

LDSET[,EPT=eptname[/...]][,ERR[=errleve]]
[,LIB[=libname]][,{MAP|HEXMAP}=[type][/lfn]]
[,NOEPT=eptname[/...]][,OMIT=ept[/...]]
[,PRESET[A][=val]][,REWIND|,NOREWIN]
[,SUBST=eptold-eptnew[/...]]
[,USE=eptname[/...]][,USEP=progname[/...]].

Controls a variety of load options; applies only to the current load sequence.

Cyber Loader RM (CDC No
60429800)
Section 2
SCOPE/HUSTLER RM
Section 7.9.2

LIBEDIT[,I=dirlfn][,L=listlfn].

Calls the utility which allows a user to define a group of central processor routines or overlays as a library.

UG Supplement: LIBEDIT
SCOPE/HUSTLER RM
Section 7.15.5

LIBLOAD,libname,eptname[,...].

Modules containing the named entry points are loaded from the specified library.

Cyber Loader RM (CDC No
60429800)
Section 2
SCOPE/HUSTLER RM
Section 7.9.3

LIBRARY[,libname][,...].

Defines the global library set; may not appear within a load sequence.

Cyber Loader RM (CDC No
60429800)
Section 7
SCOPE/HUSTLER RM
Section 7.10.1

LIMIT,n.

Resets the job's mass storage limit.

SCOPE/HUSTLER RM
Section 7.12.5

LISTTY[,I=lfn][,O=outlfn]{[,Sn][,Lm]|[,n-m]}
{[,Cn][,Wm]|[,Cn-m]}[,[n[-m]]/string/[N][U]]
[,ID=/string/][,Pn[-m]][,NR][,B][,NS][,CCx]
[,COPY][,ALL][,Z][,SAVE|CLEAR][,HELP].

Lists the file specified by I=inlfn on the file specified by O=outlfn with options for listing specified lines or ranges, listing specified column ranges, adding a carriage control to each line, etc.

SCOPE/HUSTLER RM
Section 7.6.3

LOAD,loadlfn[,...].

Loads the program(s) on file loadlfn(s) into central memory. File(s) loadlfn is not rewound before or after loading.

Cyber Loader RM (CDC No
60344200)
Section 2
SCOPE/HUSTLER RM
Section 7.9.4

MAP[,ON|,OFF|,<u>PART</u>|,FULL].

Defines the load map in effect for the remainder of a job. Interactive default is OFF, batch default is PART.

Cyber Loader RM (CDC No
60429800)
Section 2
SCOPE/HUSTLER RM
Section 7.10.2

MFL,fl.

SCOPE/HUSTLER RM
Section 7.11.2

Resets the job's Maximum Field Length.

MNF[,B[=binlfn]][,D[=char]][,E=errlev][,F][,G]
[,I[=inlfn]][,J][,K][,L[=listlfn|=0]]
[,O[={outlfn|0}]][,P][,R=n][,T][,Y][,Z].

MNF RM
SCOPE/HUSTLER RM
Section 7.16.8

Calls the MNF version 5 compiler.

MODE,n.

SCOPE/HUSTLER RM
Section 7.13.5

Selects exit conditions for arithmetic mode errors. The default condition is 7, which causes termination on all three mode errors (1, 2, and 4).

NEWNAME,oldlfn,newlfn.

SCOPE/HUSTLER RM
Section 7.7.4

Changes the name of local file oldlfn to newlfn.

NOGO[,outlfn][,eptname][,...].

Cyber Loader RM (CDC No
60429800)
Section 2
SCOPE/HUSTLER RM
Section 7.9.6

Completes loading of all program elements needed for execution, in the same manner as the EXECUTE statement, but suspends execution.

PAPERT[,B][,ID=tapeid][,O=outlfn][,T=eot].

SCOPE/HUSTLER RM
Section 7.5.11

Copies the paper tapes identified by tapeid to the disk or magnetic tape file, outlfn. The 8-bit tape frames are packed in 12-bit bytes, five bytes per word.

PFDUMP[,ADD[=addlfn[=...]]][,DROP=droplfn[=...]]
[,IX=indxlfn][,KEEP=keeplfn[=...]]
[,MT=vrn[=...]][,NEWPN=newpn]
[,NT=vrn[=...]][,O=outlfn]
[,OLDMT=vrn[=...]][,OLDNT=vrn[=...]]
[,ORDER][,U=unhdlfn][,WAIT].

SCOPE/HUSTLER RM
Section 5.4.3, 7.14.4

Allows users to dump specified permanent files from disk and/or PF dump tapes onto a new dump tape.

PFLIST[,ACCESS[=mm/dd/yy]][,ALL][,ALTERED[=mm/dd/yy]]
[ATTACH[=noatt]][,BACKUP[=U]][,BATCH]
[,DUMPED[=mm/dd/yy]][,EXPIRED[=mm/dd/yy]]
[,FDO=fdolfn][,FULL][,FULLVRN][,ID=id[=...]]
[,LASTACC[=mm/dd/yy]][,LASTALT[mm/dd/yy]]
[,LASTCAT][,LISTPN][,MT=vrn[=...]][,NT=vrn[=...]]
[,O=outlfn][,PFN=pfn][,PN=pn[=...]][,PNDEPT[=dept]]
[,PNORD=pnord[=...]][,PREFIX=prfx][,PURGED]
[,SIZE=pru][,SORT=key][,SOURCE=srce][,TTY]
[,U=unhdlfn][,VRN].

SCOPE/HUSTLER RM
Section 5.3, 7.14.5

Provides a printed report on the status of selected permanent files.

PFLOAD[,{ALL|PFN=pfn[,CY=cy]|I=inlfn[=...]}]
[,DUP=opt][,MT[=vrn][=...]]][,NT=vrn[=...]]]
[,O=outlfn][,RP=opt][,U=unhdlfn].

Causes one or more permanent files to be reloaded from a dump tape.

SCOPE/HUSTLER RM
Section 5.4.4, 7.14.6

PNPURGE[,PFN=pfn|,DPFN=dpfnd][,CY=cy][,I=inlfn].

Purges the permanent file pfn and/or the permanent files specified on file inlfn, provided that the file(s) belongs to the user. Passwords are not needed.

SCOPE/HUSTLER RM
Section 5.2.5, 7.14.7

PRINTLB,[inlfn],[outlfn].

Prints the volume header label of a SCOPE-labeled magnetic tape. The tape must have been requested as an unlabeled (Z) stranger (S) tape with nonstandard label processing (NS).

SCOPE/HUSTLER RM
Section 6.11.6, 7.6.4

PURGE,lfn.

Deletes the permanent file, lfn, from the permanent file directory; that is, it changes file lfn to an ordinary local file. The user must have attached file lfn with control permission.

SCOPE/HUSTLER RM
Section 5.2.3, 7.14.8

QU[,I=inlfn][,O=outlfn][,T=translfn]
[,TL=transleng].

Calls QUERY-UPDATE, a language for use in querying and manipulating data files organized under Cyber Record Manager with multiple indexing.

QUERY-UPDATE RM (CDC No
60498300)
Chapter 6
SCOPE/HUSTLER RM
Section 7.19.3

RANLIB[,BI=binlfn][,BO=binolfn][,D]
[,HELP][,I=inlfn][,LBI=lstblfn]
[,LD=deflfn][,LI=indxlfn]
[,LDI=indelfn|,LID=indelfn][,N=nliblfn]
[,O=outlfn][,P=oliblfn][,SL].

Calls a facility that allows users to create and modify a library of binary program decks.

SCOPE/HUSTLER RM
Section 7.15.6

REDUCE[,ON|,OFF].

Determines the execution field length of programs loaded from local files.

SCOPE/HUSTLER RM
Section 7.11.3

REPORT[,I=inlfn][,P][,R=rptname]
[,T=tablfn][,V=vallfn].

Calls the QUERY-UPDATE facility that produces reports according to specifications supplied by the user via directives.

QUERY-UPDATE RM (CDC No
60498300)
Chapter 6
SCOPE/HUSTLER RM
Section 7.19.4

REQUEST,lfn[,VRN=vrn[=...]][,NEWPN=pn]
[,MT|,LO|,HI|,HY][,RW|,RO]
[,N|,E|,Y|,Z][,NB|,IB][,NR][,NS][,S|,L].
REQUEST,lfn[,VRN=vrn[=...]][,NEWPN=pw]
[,NR][,NS][,RO|,RW][,S|,L][,AS|,EB] [,NB|,IB]
[,Y|,Z|,N|,E][,NT|,PE|,HD].

SCOPE/HUSTLER RM
Section 6.5.1, 7.4.5
(7-track)
SCOPE/HUSTLER RM
Section 6.5.1, 7.4.5
(9-track)

Requests the operator to assign file lfn. If lfn is the only parameter specified, a disk unit will be assigned; otherwise lfn will be assigned the tape set specified by vrn. If the VRN parameter is omitted but another tape parameter (e.g., RW) is specified, a scratch tape will be assigned.

RERUN[,ON|,OFF].

SCOPE/HUSTLER RM
Section 7.12.7

Specifies whether or not the job may be rerun by the operator (ON=yes).

RETURN,lfn[,...][,MT={m|SAME}][,NT={n|SAME}].

SCOPE/HUSTLER RM
Section 6.4.3, 7.4.6

Releases (detaches) local file(s) from the job, and establishes a new tape drive reservation for 7-track (MT) and 9-track (NT) magnetic tapes.

REWIND,lfn[,...].

SCOPE/HUSTLER RM
Section 7.7.2

Rewinds local file(s) lfn to beginning-of-information.

RFL,fl.

SCOPE/HUSTLER RM
Section 7.11.4

Resets the user field length to fl, but does not affect the current job field length. The job field length is raised to the user field to load user programs and, if 'AUTORFL,PART.' is in effect, to execute size-sensitive system programs.

SATISFY,[libname][,...].

Cyber Loader RM (CDC No. 60429800)
Section 2
SCOPE/HUSTLER RM
Section 7.9.7

Causes the loader to satisfy external references immediately instead of waiting until load completion; optionally specifies libraries to be used.

SCAN,[binlfn],[outlfn].

SCOPE/HUSTLER RM
Section 7.6.5

Reads file binlfn from current position to end-of-partition and lists on file outlfn information about each section encountered. SCAN was designed to identify the contents of a binary file.

SEGLOAD,[I=inlfn][,B=binlfn][,LO=listopt].

Cyber Loader RM (CDC No. 60429800)
Section 2
SCOPE/HUSTLER RM
Section 7.9.8

Initiates segment generation; must be the first statement of a load sequence.

SISTAT,islfn[,outlfn].

Cyber Record Manager (CDC No. 60499300)
Advanced Access Methods RM
SCOPE/HUSTLER RM
Section 7.18.3

Prints statistical information about an indexed-sequential file.

SKIPB,[lfn],[n],[lev],[B|C].

Repositions file lfn backwards by n sections of level lev. The C parameter must be specified for coded magnetic tapes.

SCOPE/HUSTLER RM
Section 7.7.3

SKIPF,[lfn],[n],[lev],[B|C].

Repositions file lfn forward by n sections of level lev. The C parameter must be specified for coded magnetic tapes.

SCOPE/HUSTLER RM
Section 7.7.3

SLOAD,lfn[/R|/NR],progname[,...].

Requests the loader to load the named programs from the file specified.

Cyber Loader RM (CDC No 60429800)
Section 2
SCOPE/HUSTLER RM
Section 7.9.9

SORTMRG[,I=inlfn[/R|/NR]][,MO=n][,O=outlfn[/R|NR]]
[,OWN=lfn[/R|/NR]][,6C|,7C].

Loads and executes the SORT/MERGE utility. The file specified by I=inlfn contains directives specifying the files to be sorted and/or merged, the record structure, sort keys, etc.

SORT/MERGE v4 RM (CDC No. 60497500)
SCOPE/HUSTLER RM
Section 7.5.12

SWITCH,n[,ON|,OFF].

Sets program switch n to a new position. The switch may be toggled or set to a specific position.

SCOPE/HUSTLER RM
Section 7.12.8

TAPRES,{MT={m|SAME}|NT={n|SAME}}.

Sets the job 7-track tape reservation count to m, and the 9-track reservation to n.

SCOPE/HUSTLER RM
Section 6.4.2, 7.4.7

TRAP[,I=inlfn][,O=outlfn].

Loads the TRAP routine, which activates the DEBUG Aids execution time routine TRAPPER; applicable to the next relocatable load sequence.

Cyber Loader RM (CDC No 60429800)
Section 6
SCOPE/HUSTLER RM
Section 7.13.6

UNLOAD,lfn[,...].

Rewinds and unloads magnetic tapes and releases the local files from the job.

SCOPE/HUSTLER RM
Section 7.4.8

UPDATE[,A][,B][,C=complfn][,D][,E][,F|,Q]
[,G=pmodlfn][,I=inlfn][,K=complfn]
[,L=opt][,M=mrglfn][,N=npllfn]
[,O=outlfn][,P=opllfn][,R=opt]
[,S=srclfn][,T=sxclfn][,U][,W][,X]
[,Z][,8][*=char][/=char].

Calls the UPDATE utility to create, retrieve, or modify decks of text cards on the specified program library.

UPDATE RM
(CDC NO. 60449900)
SCOPE/HUSTLER RM
Section 7.15.7

Index

Comment Sheet

TITLE: SCOPE/HUSTLER Reference Manual

REVISION: H

The MSU Computer Laboratory solicits your comments about this manual with a view to improving its usefulness in later editions.

For what applications do you use this manual?

Do you find it adequate for your purposes?

What improvements do you recommend to better serve your purposes?

Note specific errors discovered (please include page number reference).

General Comments:

FROM

Name: ____

Department: ____

Address: ____

Michigan State University
User Information Center
Computer Laboratory
East Lansing, Michigan 48824

Revision Request Form

TITLE: SCOPE/HUSTLER Reference Manual

REVISION: H

To automatically receive the next free revision packet for this manual, complete and return this form within 30 days of purchase to the User Information Center, Computer Laboratory, Michigan State University, East Lansing, Michigan 48824.

This form must be submitted even if you have received past revisions for this manual.

Name: ______________________________

Department: ______________________________

Address: ______________________________

Date of Purchase: ______________________________

No postage necessary if sent through campus mail. Fold on dotted lines and staple.

Michigan State University
User Information Center
Computer Laboratory
East Lansing, Michigan 48824