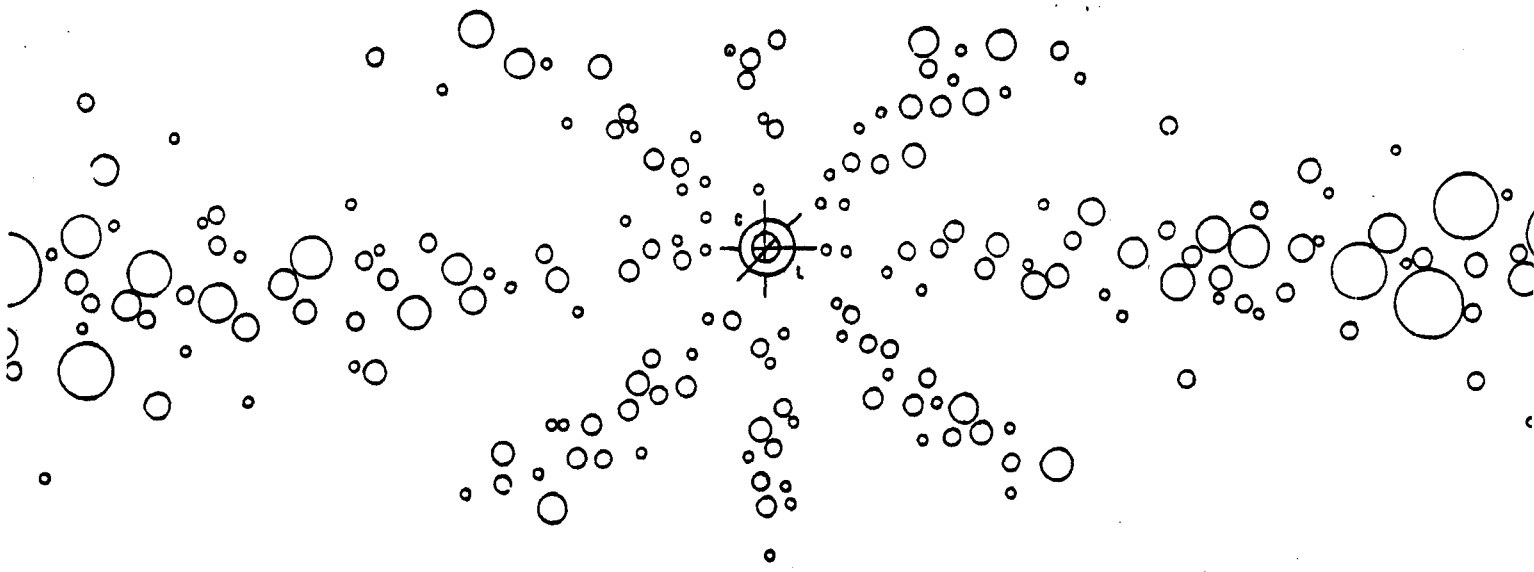


The Interactive System
User's Guide

Computer Laboratory

Michigan State University

REVISION RECORD	
Revision	**Description**
	Original printing. (9/72)
A	Add descriptions of EXEC, special editing directives,
	AUTOREP, CONSULT. Add Appendices J and K. Update
	Authorization File utility descriptions. (1/74)
B	New descriptions of EDITOR corresponding to Version 2.
	EDITOR error messages updated. (7/75)
C	Retitle to "Interactive System User's Guide." Delete
	obsolete information and material not unique to interactive
	system. Add Chapter 8 (Front-End Commands) and Chapter 9
	(EXEC Files). Add description of character sets in Chapter 5.
	Add descriptions of AUTHORF and paper tape utilities in
	Chapter 2. Add log-in options in Chapter 1. Add Appendices
	B and C. Update all other material except Chapter 3
	(EDITOR), Chapter 6 (Debugging Aids) and Appendix H (Sample
	Terminal Sessions). (1/78)
D	Corrections to Revision C. Update Appendix A. (3/78)
E	Addition of ASCII graphics - character set conversion.
	Reorganization and typesetting of Appendices D and G. (12/78)
F	Chapter 3 - EDITOR rewritten. Addition of new material for
	ASCII printers. Replacement of Appendix H. (5/79)
G	Chapter 6 - Debugging rewritten. Information on auto-baud
	service added to Chapter 1.

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**
G(cont)	Add new Front-End command to Chapter 8. Appendix D Revised
	and EDITOR section typeset. Miscellaneous revisions and
	corrections made. (8/79)
H	Chapter 1 - Authorization, Logging In and Logging Out, Chapter
	7 - DISPOSE--Routing Files for Off-Line Processing and
	Chapter 8 - Front-End Control Characters and Commands
	rewritten to include new terminology and more examples. The
	Front-End command, %LOGINMSG, and new default control charac-
	ters added. Miscellaneous revisions and corrections made.(12/79)
J	Chapter 2 - System Commands and Editing Directives and
	Chapter 3 - EDITOR--The Text Editing System rewritten to
	include new commands and examples. SPSS example added
	to Appendix H. (5/80)
K	Miscellaneous changes and corrections made, reflecting
	recent changes to the operating system. (2/81)

Note: Revision letters I, O, and X are omitted to avoid confusion.

Preface

The *Interactive System User's Guide* describes all commands and procedures that specifically relate to the use of the interactive system on the MSU computer system. It is the primary reference for the interactive text editor, the front-end computer system, and certain SCOPE/HUSTLER commands that are available only via interactive access.

Some commands have different functions in batch and interactive modes; this user's guide is concerned solely with interactive use of these commands. Users requiring more information about specific system commands, as well as a general overview of the SCOPE/HUSTLER operating system, should refer to the*SCOPE/HUSTLER Reference Manual*.

Other sources of information include: pocket guides and HowTos. Several HowTos deal specifically with the use of the interactive system, these handouts give a step by step description of several basic computing tasks, with examples and illustrations.

A current list of other publications which may be applicable to the interactive use of the MSU system can be obtained from the User Information Center. In addition to published documentation, on-line assistance is provided by the routine HELP, which can produce at a terminal brief descriptions of any interactive utilities (see Section 2.9.1).

The original version of this publication was released under the name Computer Laboratory User's Guide Volume IV, "The MSU Interactive Computing Facility," issued in May, 1972. This first edition was written by James A. Lukey and edited by Leonard H. Weiner. The current edition represents a major rewriting of the original text.

Tory A. Sawyer
Deborah A. Alpert
Dianne M. Smock

Publications Office
User Information Center
Computer Laboratory

Table of Contents

APPENDICES

Introduction

The interactive computing facility at Michigan State University uses the SCOPE/HUSTLER operating system which itself is a locally developed extension of the CDC SCOPE and NOS/BE operating systems.

You may communicate with the MSU computer system via normal telephone lines from an ASCII terminal that transmits data at 10, 30, or 120 characters per second. From the terminal the interactive system user can issue any command available to the batch user, except those commands that involve magnetic tapes. This one restriction can be overcome, however, by submitting a job file from the terminal for batch processing. In addition, you have text editing and interactive I/O capabilities that facilitate tasks such as program debugging, where instant feedback is especially valuable.

The format of this volume was designed with both the new and the experienced user in mind. To provide a coherent manual suitable for self-instruction, the descriptions of the various commands and directives are grouped according to function, and each chapter and major section opens with a short paragraph which describes the material contained in the immediately following subsections. In addition, many examples have been included to illustrate the use of relatively complex directives and commands. (See Appendix G for a summary of all interactive commands and directives.)

The chapters have been arranged in the following manner:

Chapters 1 and 2 discuss basic procedures and essential background material for using the MSU computing system interactively, such as rules for user input, the difference between commands and directives, the log-in/log-out procedures, and commands unique to the interactive service.

Chapter 3, the largest, deals with EDITOR, the text editor that enables you to construct and edit (on-line) source programs, data, or any other type of text. It also describes how to compile source programs created or edited under EDITOR. The early sections of the chapter should give you enough information to begin to use EDITOR. As you become more familiar with EDITOR, you can acquire more sophisticated techniques included in the later sections of the chapter.

Chapter 4 is reserved for future descriptions of software.

Chapter 5 describes procedures for interactive communication with an executing program. Chapter 6 describes interactive use of SCOPE/HUSTLER debugging aids.

Chapter 7 describes the DISPOSE command, which enables you to print or punch files on remote I/O devices, and to submit batch jobs from the terminal.

Chapter 8 describes the control characters and commands processed by the Front-End computer which acts as an interface between interactive terminals and the main computer.

Chapter 9 describes the use of the exec file: a mechanism which allows you to encapsulate frequently used control statement sequences and to automatically initiate a control statement sequence upon successfully logging in.

This volume does not attempt to duplicate information which is easily obtainable from other sources, such as Control Data publications. A current list of publications applicable to the MSU system can be obtained from the User Information Center. On-line assistance is provided by the program HELP, which contains brief descriptions of all interactive utilities (see Section 2.9.1).

Some examples in this manual illustrate statements that a user enters at the terminal and the output that results. When an example contains both input and output, the input lines are shaded to distinguish between input and output. Note: In some cases, output examples have been slightly reformatted to save space. When the example requires the use of a non-printing character, the character function code will be boxed and shaded (e.g. LITERAL).

The following conventions are followed when describing the format of interactive commands.

UPPER CASE	item must appear as shown
lower case	item must be supplied by user
|	separates alternate forms
{ }	encloses alternate forms
[ ]	encloses optional forms
———	underscores default form
= = = =	underscores abbreviation

1

Authorization, Logging In and Logging Out

The MSU computer system identifies and charges each of its users by problem number (PN) and user ID. The PN is an account number, and since there may be more than one user assigned to a single PN, the user ID enables the system to keep track of each individual subaccount. When you receive a PN, user ID, and password, you are assigned certain limits on the amount of system resources available to your account. All of this information is kept by the system in a permanent file called the Authorization File. While the average batch user may not need to use the password, the interactive system user must use all three of the identifiers (PN, user ID, and password) in order to log in successfully.

Note: Users of the interactive system are charged for computing services at Rate Group 3 from 8:00 a.m. to 11:00 p.m. This means that you pay 150% of the normal charge for computing resources for interactive service. Rate Group 1 rates will be available from 11:00 p.m. to 7:00 a.m. (You pay only 50% of the normal charge). If you begin during RG3 hours, you will be charged RG3 rates for the entire session.

1.1 Logging In

You must have an ASCII terminal to communicate with the MSU computer system. The terminal may be hard-copy or cathode ray tube (CRT). Hard-copy terminals produce printed paper output while CRTs display results on a TV-like screen. The dial-in procedure varies slightly among the different types of terminals; most are used with an ordinary telephone, which is connected to the terminal by an acoustic coupler. Terminals are designed to transmit a certain number of characters per second to and from the computer. The transmission rate is called the baud rate; the three most frequently used baud rates are 110 (10 characters per second), 300 (30 characters per second) and 1200 (120 characters per second). Many terminals will allow any of these baud rates so that you can choose between the three speeds. All interactive phone lines at MSU have an automatic baud rate detection capability.

For 110-300 baud lines dial 353-8500 to connect with the MSU computer system. When using an MSU campus phone, only the last five digits of the number (38500) should be dialed. The phone number for the dial-up automatic 110-300-1200 baud lines is 353-8570.

If the computer is in operation you will hear a high-pitched carrier tone on the telephone line when the phone is answered; otherwise, you will receive a busy signal or a recorded explanatory message. When you hear the carrier tone place the telephone handset firmly into the terminal coupler receptacle cups with the phone cord at the designated end. The computer will automatically assume a baud rate of 300 unless you enter a carriage return or CTRL-X[1] which allows the computer to detect the actual terminal baud rate (which may be 110, 300 or 1200). At this time, if the terminal is properly connected, a header line will be displayed at the terminal and authorization information will be requested:

[1]The carriage return and CTRL-X are the default characters for the end-of-line (EOL) and line delete (CANCEL) characters, respectively. These and other special characters may be redefined by the user during the interactive session. See Chapter 8.

```
  11:24:39 01/16/81  MSU-FREND   04.13   SOCKET= 55
CPORT  273
01/16/81  MSU HUSTLER 2       LSD 50.26  01/13/81  CYBER750

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

The header contains information about the current operating system and your interactive connection, including the date and time. Respond to the authorization request within five minutes by typing the proper information over the blacked out spaces in the forms detailed in Sections 1.1.1 and 1.1.2. You can also use the ECHO character (default is CTRL-V) to turn off echo printing while entering your password. (This will prevent the display of the password on a CRT.) See Section 8.3.6.

If you do not enter the correct information within five minutes, a message indicating your error will be printed and the log-in procedure is reinitiated. You are allowed three attempts to log in successfully after which, if you are still unsuccessful, the system will provide the following display and disconnect the terminal phone connection.

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

Note: The transmission of a line of data to the computer is completed when the end-of-line is depressed. The default EOL character is the carriage return; thus, a carriage return must end every typed line.

1.1.1 Normal Log-in Procedure

After the system presents the request for logging in, enter the requested information.

```
password,problem-number,user-id
```

This line is terminated by depressing the carriage return key.

When the information has been correctly entered, the system will respond with a number of messages similar to those below:

```
                                                    Notes
SS22522, USER 7 (P012, S043) RGn                    1
LAST ACCESS: S 04/18/77 16:05                       2
RUNS: 21 BALANCE: $ 4945.16                         3
FORTRAN, 20 LINES, LENGTH 72.                       4
READY .13.05.37.                                    5
```

Notes:

1. SS22522 is your unique sequence number for this session.

   User 7 is a number used in an internal system table.

   P012 is the port number in the Front-End computer.

   S043 is the socket number, which corresponds to the interactive phone line.

   RGn is the rate group (dependent on log-in time; see Section 1.1).

2. Source, date and time of last access to the MSU computer system for this user-id.

3. Total number of computer system accesses by this user ID since the user ID was authorized, and its current dollar balance.

4. Status of the text editor (see EDSTAT directive, Section 3.17). This message only appears if a non-permanent EDITOR work file exists from a previous session.

5. The READY message appears with the current time of day (on a 24 hour clock) to indicate that the system is ready to receive a command.

1.1.2 Shortened Log-in Procedure

An abbreviated form of this display may be obtained if you add a comma and an 'S' to your response to the log-in request, i.e.,

password,problem-number,user-id,S

Again this line is terminated by depressing the carriage return key.

In this case, the system will respond with:

```
SS22522, USER  7        (S  43, P 12)  RGn
READY .13.05.37.
```

1.1.3 Special Auto-Exec Log-in Procedures

An initialization file (init file) is a program or sequence of commands to be executed at log-in time. This may contain frequently used procedures in order to save time for the user. This file is created and changed only by the PN manager, who controls its use with the Authorization File. The PN manager establishes an automatic execution procedure (Auto-Exec) controlling the use of the init file. Use of the INIT file may be optional or required.

If an optional initialization file has been established for your problem number, you may request the initialization file as follows:

password,problem-number,user-id,INIT[,S]

You may bypass normal execution of the initialization file by specifying

password,problem-number,user-id,NOINIT[,S]

See Chapter 9 for a description of the automatic execution procedure.

Note: The initialization parameters (INIT,NOINIT) and the 'S' parameter may be specified in either order.

1.1.4
Obtaining an Additional Interactive Connection

Once you are logged-in to the computer, you may obtain an additional interactive connection in order to run two jobs simultaneously from your terminal. Use the Front-End command "%LOGIN." The %LOGIN command produces a prompt for log-in information. This is fully documented in Chapter 8. You must use two different accounts to have simultaneous connections.

1.1.5
Connect Time

Connect time is the amount of time, in minutes, that you are allowed for one interactive session. After logging out, you may immediately log in for another session.

The default connect time limit is 60 minutes. Connect Time, abbreviated CT, is a field on your authorization file. This limit may be changed only by the PN manager. (See Section 2.6.2 in the *SCOPE/HUSTLER Reference Manual*.)

Since RG1 rates terminate at 7:00 a.m., the connect time will be adjusted to the smaller of:

your authorized connect time, or

the time remaining until 7:00 a.m.

If your connect time is set to less than your authorized limit, you will receive the message:

```
YOUR CONNECT TIME WAS REDUCED TO N HOURS AND NN MINUTES;
RG1  SERVICE  ENDS  AT  .07.00.00.
```

1.2
Logging Out

Logging-out means terminating your interactive session. This can be accomplished in several ways. The normal method uses the LOGOUT command. You can also terminate a session by hanging up the phone. If you are connected to the computer for the maximum amount of time specified in the Authorization File, the system will automatically disconnect you because you have reached your "connect time" limit. In addition, there is a Front-End command, %QUIT, which is identical to hanging up the phone. It is most useful when you are running two jobs from one terminal and wish to terminate one and not the other.

1.2.1
Normal Termination

Normal termination of an interactive session begins when you type:

```
LOGOUT.
```

The LOGOUT procedure will automatically return all permanent files, all empty files, and all local files with names beginning with 'ZZZZ' to the system. The permanent files will retain their permanent status. The 'ZZZZ' and empty files, however, will be destroyed. You will then be asked to type a disposition for all other local files. The system displays one local file name (followed by a question mark) at a time. You are first asked about the disposition of the EDITOR workfile (unless it is empty or permanent). Enter one of the following disposition codes for each file.

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, under the user ID/problem number combination which was used to retain them, for two hours after LOGOUT. 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:

Sample:

```
CP USE        2.625 SEC              VALUE $  .11
PP USE       58.409 SEC              VALUE $  .19
CM USE        1.731 W-H              VALUE $  .45
CT USE        .340 HRS               VALUE $  .85
TOTAL VALUE OF JOB AT RG3 $                   1.98
```

If you wish to drop all local files and do not want detailed accounting information on your interactive session, type 'LOGOUT,T.' In this case, the following message is given and the terminal is immediately disconnected.

Sample:

```
LOGOUT,T.

JOB COST: $ 1.98
```

1.2.2 Hanging Up or Accidental Termination

If you hang up the phone, either accidentally or intentionally, your session is terminated. Similarly, if the computer breaks the connection due to hardware or software problems, your session is terminated. But all is not lost. Your local files will be retained for two hours. Thus, you are given time to log in again, under the same user ID and problem number, to continue the interactive session.

1.2.3 Automatic Logout

A connect time limit, in real-time, is established for each problem number at the time that computer authorization is obtained. This limit is enforced for each terminal session. When this time limit has expired, the system will initiate a LOGOUT for the user. However, the system issues a warning message five minutes before this time, and again two minutes before the automatic LOGOUT.

1.2.4 Using %QUIT

%QUIT is a Front-End command which acts the same as hanging up the phone. It is commonly used when you have two simultaneous connections. You can use %QUIT to terminate one connection while retaining the other. It may be used on a single connection as well. As with hanging up, all local files are retained for two hours and no accounting information is displayed. This command is fully described in Chapter 8.

2
System Commands and Editing Directives

2.1
Communicating with the Interactive System

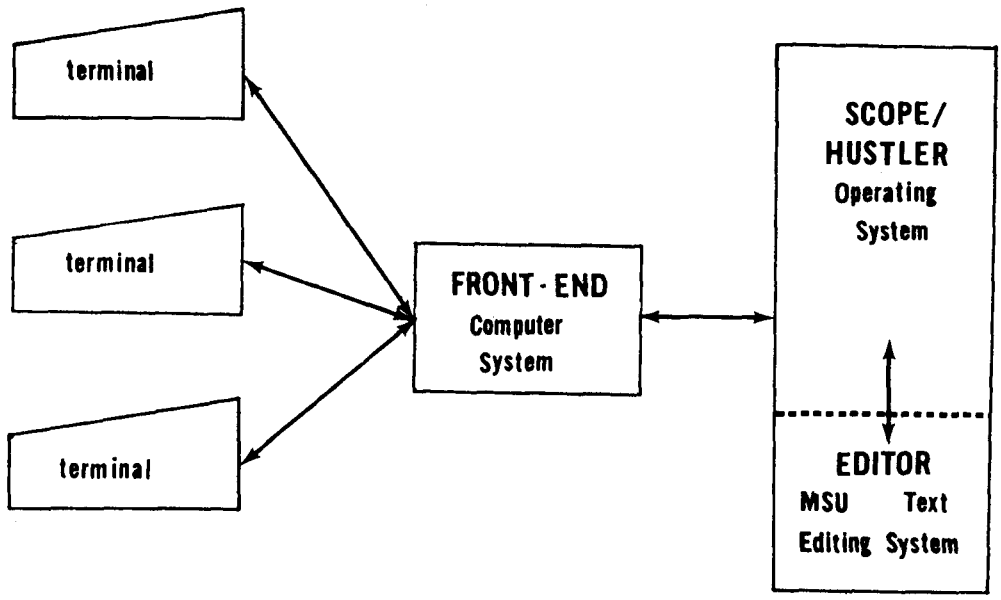
The MSU interactive system is composed of:

- the SCOPE/HUSTLER operating system, which allocates computing resources, contains many commonly used programs, performs accounting functions, etc.
- the Front-End computer system, which controls communication between terminals and the main computer.
- EDITOR, the text editing system.

As an interactive user, you can communicate with the MSU interactive system in three ways:

- system commands
- Front-End commands
- EDITOR directives

Consider the following simplified diagram of how the interactive system works.



The Front-End handles communication between terminals and the main computer. It can store input from all terminals connected to the system and transmits information to the operating system when SCOPE/HUSTLER has the time and resources to efficiently process the instructions. The instructions you type at the terminal are sent to the Front-End computer system which either acts on the command if it is a Front-End command, or transmits the command to the operating system. The Front-End directly processes control characters which affect terminal function and Front-End commands which cover a number of interactive functions. Other instructions are transmitted to the operating system.

The operating system determines whether the instruction is a system command or an EDITOR directive. The EDITOR program is part of the SCOPE/HUSTLER operating system which supplies interactive users with an important tool for developing files of information and programs for computer processing. EDITOR directives are instructions for the EDITOR program.

The first section of this chapter presents the general formats of system commands, Front-End commands and EDITOR directives. The remainder of this chapter and chapters 5, 6, and 7 describe specific system commands. Chapter 3 is devoted to editing directives, and Chapter 8 to Front-End commands.

2.1.1 System Commands

The interactive system commands include all of the standard SCOPE/HUSTLER control statements available to the general batch user on the MSU computer system, with the exception of magnetic tape commands. However, because of the special nature of interactive use, some of these commands have a somewhat different meaning to the interactive user, or have special limitations. In addition, there are some interactive commands which have no batch control statement equivalents.

System commands may be entered one at a time or in strings, i.e., several commands on a single line. The following rules apply to interactive commands.

1. More than one command may be entered on a line, but each command must be terminated by a period or right parenthesis. The next command may begin immediately on the same line. No period or right parenthesis is needed for the last command on a line (unless otherwise noted).

2. Parameters are separated by commas. Blanks are ignored (unless otherwise noted).

3. An end-of-line character (the default is carriage return) terminates the line and initiates sequential execution of the commands. Commands may not be continued onto the next line (unless otherwise noted).

Example:

The following system command string might be entered at the terminal:

```
ATTACH,BOB,PFBFILE.COPYCF(BOB,XZ)DISPOSE,XZ,PR.
```

The ATTACH command will be executed first, followed by the COPYCF command and the DISPOSE command.

In the event of an abort during the processing of a command string, processing will skip to the next command following an EXIT command. If no EXIT command is present, execution of the command string will terminate. The remaining commands on the line will not be executed. (See *SCOPE/HUSTLER Reference Manual*, section 7.1.3 for a description of EXIT.)

Interactive-only Commands

The following commands are unique to the interactive system and are referred to as interactive commands. Each is described in this volume under the section number listed in parentheses.

DEBUG	enables Cyber Interactive Debug facility (6.3.5)
EOT	signals the end of a paper tape being read in via TAPE or TAPEC. (3.7.4)
LISTAPE	lists all tapes waiting to be mounted. (2.9.5)
LOCK	locks out messages from other users. (2.4.3)
LOGOUT	terminates the interactive session. (1.2)
MESSAGE	sends messages to the operator. (2.4.4)
N	initiates automatic line numbering of text lines. (3.7.1)
OK	changes the standard response to 'OK-'. (2.3)
PROMPT	initiates an asterisk prompt when input is expected from the terminal. (5.5)
READPT	designates that a paper tape is to be read and copied to a file. (2.8.1)
READY	changes the standard response to 'READY .hh.mm.ss'. (2.3)
RTL	sets the execution time limit. (2.5.1)
SEND	sends messages to other users. (2.4.2)
SITUATE	lists the user IDs of other users logged in. (2.4.1)
TPREAD	designates that a paper tape or another form of auxiliary storage is to be read and copied to a file. (2.8.1)
WRITEPT	writes the contents of a local file on a paper tape. (2.8.2)

Other commands which are not unique to the interactive system but which have slightly different functions or are of particular interest to interactive users include:

CONNECT	connects a file to the terminal. (5.2)
DISCONT	disconnects a file from the terminal. (5.4)
*EOS[nn] (*EOR[nn])	inserts an end-of-section. (2.2)
*EOP (*EOF)	inserts an end-of-partition. (2.2)

LISTTY lists the contents of a Display code file at the terminal. (2.6.2)

PAGE lists the contents of a Display code file at the terminal, page by page. This is most useful on CRT terminals. (2.6.3)

SETCODE sets the character code associated with a local file. (5.3)

2.1.2 Front-end Commands

The Front-End computer system allows you to control input and output from your terminal and facilitates the use of a wide variety of terminals. Front-end commands are not transmitted to the main computer but are processed by the Front-End computer, which acts as an interface between you and the main computer. The action designated by a command is taken immediately.

The following rules apply to Front-End commands:

1. Front-End commands begin with the Front-End control character; the default is the percent sign (%). This distinguishes Front-End commands from system commands, EDITOR directives and other program input.

2. Front-End commands are terminated by an end-of-line character. the default is the carriage return. As a result, only one Front-End command may appear on a line.

Since these commands are accepted directly by the Front-End computer, they do not interfere with your job on the main computer. You may enter Front-End commands at any time.

Each Front-End command is described in Chapter 8.

2.1.3 Editor Directives

Text editing directives give you the ability to build and edit text and program files interactively. The text editing facility is called EDITOR, a full description of which is found in Chapter 3.

EDITOR directives must conform to the following rules:

1. More than one directive may be entered on a line, but each directive must be terminated by a period. The next directive may begin immediately on the same line. No period is needed for the last directive on a line.

2. Parameters are separated by blanks or commas.

3. An end-of-line character (the default is carriage return) terminates the line and initiates sequential execution of the directives. Directives may not be continued onto the next line.

The same formatting rules apply to EDITOR directives as for system commands, except that blanks are legal delimiters between parameters. EDITOR directives may be entered one at a time, in strings, or interspersed with system commands.

EDITOR Text Lines

You may build a text file by entering text lines directly at the terminal. Text lines must start with a line number, and may be entered whenever the system has responded with the READY or OK message. The receipt of a text line is acknowledged by sending a line feed back to the terminal. You may then enter another text line or any command or editing directive.

Numbered text lines are actually a special class of EDITOR directives, since they instruct the text editor to add the text line to the current file being built or edited (see Chapter 3).

2.1.4
Distinguishing between System Commands and Directives

Although the operating system discriminates between system commands and EDITOR directives, some ambiguity may exist because some commands and directives use the same flagword (e.g. FTN, COBOL, BASIC). Therefore, the following conventions have been established:

1. If a line begins with a dollar sign ($) or a plus (+), what follows is treated as a system command.

   Example: FTN. normally is an EDITOR directive.

   $FTN. specifically designates 'FTN.' as a system command. However, any existing user file or program would be executed in preference to the actual system command. (See the 'name' call statement, Chapter 7 of the *SCOPE/HUSTLER Reference Manual.*)

   +FTN. designates 'FTN.' as a system command and causes the system program to be executed in preference to a user program. (See the 'name' call statement, Chapter 7 of the *SCOPE/HUSTLER Reference Manual*).

2. The following compiler names are considered editing directives if they are immediately terminated by a period or a blank, but are considered system commands if they are followed by a comma or left parenthesis.

   BASIC COBOL FTN

   Example: FTN. is an EDITOR directive.

   FTN,I=COMPILE. is a system command.

3. Any line that begins with a hyphen (dash or minus sign) is treated as an editing directive.

   Example: FTN,I=ABC. is a system command.

   -FTN,150-300. is a text editing directive.

2.2
End-Of-Section and End-Of-Partition

The special entries *EOS and *EOP signify end-of-section and end-of-partition, respectively, to an interactive program that is reading input entered at the terminal. End-of-section may also be indicated by *EOSnn, where nn is the octal level number ($0 \leq nn \leq 17$). If nn is omitted, level 0 is assumed. NOTE: The carriage return must immediately follow *EOSnn or *EOP, or the entries will be treated as normal data rather than as special directives.

You can also enter *EOS and *EOP as numbered text lines, which will be inserted into the file being constructed under EDITOR. When the contents of that file are transformed into a standard coded file, the special text lines *EOS and *EOP are converted to SCOPE end-of-section and end-of-partition marks, respectively.

For compatibility with earlier systems, *EOR, *EORnn and *EOF have have been retained as valid entries: *EOR signifies end-of-section, *EOF signifies end-of-partition.

2.3
System Prompts — READY and OK

After you enter a line of directives or commands, the interactive system indicates that execution has been completed by transmitting a prompt for input to the terminal. The default is a READY message, which includes:

1. a carriage return and line feed, followed by
2. READY.hh.mm.ss. (the time in hours, minutes, and seconds), followed by
3. a carriage return and one line feed.

```
READY 13.19.03
```

If you prefer a briefer response, type: OK. Thereafter, the system will respond: 'OK-' without a following line feed or carriage return. To return to the READY message, the command is: READY.

```
READY 13.19.38
ok.
OK-
```

2.4
Inter-Terminal Communication

Interactive system users can communicate with one another through messages displayed at their terminals. The following commands show you how to identify other users, send messages, and prevent the receipt of messages from other users.

2.4.1
Who is Logged in? — SITUATE

If you need to communicate with another user, you first must know if that user is currently logged in on the system. Use SITUATE to determine which other users are, in fact, logged in. Type:

SITUATE.

The system will respond with a list of all currently logged-in user ids.

```
READY 13.20.31
situate
OPERATOR        KELLY           HAZARD          BOB
POWERS          KERN            HAL             SZKOT
TURNBULL        STATDISP        MESSE           IOROOM
KUSHLER         STEMBOL         ENCOUNTRJS      V1000
FIB803JL        CEM             CRS             AOUATE
YEH             ANDERSON        HOSKIN          ENCBOGER
PRAWAT          NEITZ           REID            MUGWUMP
SZKOT1          DAN             SHOEJWG         SMITH
SCL             WHITE           ELECTROLAD      HELFER
LEWIS           REDACT2         LARSON          PWP
INGVALDSON      CHRISTENSE      BUTLER          WARD
ECHOFL2         MARSHALL        TROSKO          SIN
SPYKE           LACY            GUENON          CEM131
MAINTOW1        KALES           BUTLER          ENA870JJ
OFFDAILY        SHAPIRO         TUCKER          HARRISON
ELAINE          BROWN           TOM             SAFIR
```

2.4.2
Sending a Message — SEND

To initiate communications with any user who is currently logged in, type:

SEND,id.

The id typed after SEND should, of course, be one of those found in the list produced by SITUATE. If the user is not found, the system responds with the following message, prompting for a new user id:

```
I COULD NOT FIND THIS USER
TO WHOM-
```

The message to be sent is prompted by

TYPE MSG OR END*

When you wish to terminate SEND type END alone on a line.

No message line should contain over 50 characters as any excess characters are not sent.

```
READY 13.20.38
send.
TO WHOM- redact2
TYPE MSG OR END *hi vicki...what's new?
```

If you are on the receiving end of a message you will see the message prefixed with

MSG FROM id - message

This gives the user id of the person sending the message.

```
MSG FROM ELAINE     -HI VICKI...WHAT'S NEW?
```

2.4.3 Inhibiting Messages — LOCK

You may wish to avoid unsolicited messages. The LOCK command prevents your terminal from printing messages. This is particularly desirable when a final copy of a program result is being generated, and you do not want to receive spurious messages that have been sent by other users or the operator.

LOCK[,ON|,OFF|,PART].

LOCK,OFF. This is the default (normal) condition which allows any message to be received by the terminal.

LOCK,PART. This will lock out all messages from other users. However, messages from the operator will not be inhibited. If you enter LOCK with no parameters, PART is assumed.

LOCK,ON. This will lock out all but the most urgent messages from the operator.

```
READY 13.25.31
lockon.
  :
send
TO WHOM- elaine
THIS USER IS LOCKED OUT
TO WHOM- end
```

2.4.4 Sending a Message to the Operator — MESSAGE

The MESSAGE command is used to communicate with the operator. Type:

MESSAGE.

You are then prompted for the message to be sent to the operator's terminal located next to the main computer console. Again, the message is terminated by typing END alone on a line.

As the Operations staff is extremely busy, response to a message should not be expected. MESSAGE should be used primarily to inform an operator of an unusual situation. Operators will not answer programming questions.

2.5
Controlling System Resources

You can control your use of several system resources. For the most part, batch control statements and interactive commands for such controls are identical, with one exception, RTL, described below. Any batch control statements concerned with magnetic tape resources are illegal in interactive usage, since tapes are not accessible to interactive jobs.

2.5.1
Setting Command Time Limit — RTL

In batch processing you must specify the maximum amount of central processor (CP) time that will be used by the entire job; this is done with the T parameter on the job card. In an interactive job, the overall time limit is set to the maximum allowed in the Authorization File. An additional limit exists on the amount of CP time that can be used by any individual command. The default is 7 seconds. To reset this limit for an interactive session the RTL command must be used. (Note that this limit is different from "connect time." See Section 1.2.3.)

RTL,nnn.

If you wish to change the time limit, the value nnn must specify the new command time limit in octal (base 8) seconds. The value given must not exceed the maximum command time limit imposed by the Authorization File, minus the amount of CP time already consumed by this interactive job. (You may list your Authorization File limit by typing 'ASSETS' or 'AUTHORF,DISPLAY,TIME').

If a command or user program exceeds the command time limit set then the system will display the message

TIME LIMIT

To remedy, merely type 'RTL' with a sufficiently large value and re-execute the command that was terminated. Use caution in extending RTL. Exceeding the command time limit may indicate a problem in your program.

```
OK-[redacted]
OK-
```

2.6
File Commands

Supplementing the standard SCOPE/HUSTLER commands that manipulate files are three utilities that are especially useful to interactive users. (See the *SCOPE/HUSTLER Reference Manual*, section 7.7 for other file manipulation utilities.)

2.6.1
Listing the Names of your Local Files — FILES

The FILES command lists the local files associated with your current interactive session.

FILES[,O=lfn].

O=lfn specifies an optional output file. TTYTTY is the default file for interactive jobs when lfn is not given. This normally causes output to be listed at your terminal. If the O parameter appears alone, output will be on the connected file ZZZZOT.

Sample Output:

```
OK-files.

C*OUTPUT    P*DATA     TTYTTY    OUTFILE

P*INITFIL   C*INPUT
```

where attached permanent files are prefixed by P* and connected files by C*. Files are listed four per line, using as many lines as needed.

2.6.2 Listing the Contents of a File at the Terminal — LISTTY

LISTTY lists at the terminal the contents of a coded file. LISTTY is called as follows:

LISTTY[,p_1,p_2,...,p_n].

The parameters for LISTTY may appear in any order. Legal delimiters are the comma and left parenthesis; terminators are the period and right parenthesis. The following parameters are recognized.

Input/Output

I=inlfn	specifies the input file. The default input file is FILE.
O=listlfn	specifies the output file name. In interactive use, 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 the terminal.Note: If O=OUTPUT, a page eject will precede the output unless the COPY parameter is used. If the file is listed at the terminal, one line feed precedes the output.
B	suppress printing of extra blanks. Occurrences of two or more consecutive blank characters are reduced to a single blank. This option also selects an input line width of 137.
NS	omit line numbers in the output.
CCx	use x as the carriage control for each line (e.g., CC0 for double space). The default carriage control is a blank.
COPY	suppress the addition of the carriage control and the printing of the line numbers. This option also selects ALL.
ALL	list all lines selected by the other options. This option suppresses the skipping of duplicate lines after the first has been printed.
Z	suppress printing of the end-of-section marks.

SAVE	save the parameters of this LISTTY call on file ZZZZDFS. Subsequent calls to LISTTY within the same terminal session will be executed as if these parameters were included among those appearing in the later calls. Including this option will suppress any listing; that is , only the SAVE operation is performed.
CLEAR	return file ZZZZDFS, which clears the saved parameters before the other parameters of the current command are processed.
HELP	print a full help listing, explaining how to use LISTTY.
File Positioning	
Sn	list the file starting at the LISTTY line number n (default is 1). Up to 20 line ranges may be specified in ascending order (either by Sn and Lm, or n-m).
Lm	stop listing the file at line m. Up to 20 line ranges may be specified in ascending order (either by Sn and Lm, or n-m). If m is 0, list to end-of-information (which is the default).
n-m	same as Sn,Lm. Up to 20 line ranges may be specified in ascending order (either by Sn and Lm, or n-m).
Cn	list each line starting at column n (default is 1, maximum is 137).
Wm	list up to and including column m of each line. The default column width is normally 72 for interactive jobs, but is changed to 137 if the B or O=listlfn parameters are selected. 137 is the maximum value allowed.
Cn-m	same as Cn,Wm.
Rn	list n sections. If n is 0 or n is omitted, list one section.
Pn[-m]	specifies a page number or page range to be listed. This parameter may appear up to 10 times, in ascending order of page numbers. 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.
NR	list the input file from its present position (i.e., no rewind).
Search Strings	
[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. The N, U, and column range parameters modify the string search in the same way as in EDITOR text search strings (see Section 3.4.3). Note, however, the different placement of the column range numbers.

The string may be up to 50 characters. A slash within the string must be represented by two slashes (//). A period or right parenthesis must be represented by a slash followed by the octal display code (e.g., /57 and /52). Other characters may be represented in this manner, but it is only required for these two because they would otherwise be recognized as terminators.

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 in column one of each line. This option is useful for locating a program listing within the input file.

Default Parameters

If no parameters are given, the following will occur:

LISTTY will list the contents of the local file named FILE on the system file TTYTTY, which normally causes the output to be listed at your terminal, and will output only columns 1-72 of each line.

Each line will be prefixed by a blank (the carriage control) and a four-column line number generated by LISTTY followed by a blank. (If the line number is more than four digits long, only the last four digits will be displayed.)

A blank line with a '1' carriage control will precede the first line listed.

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 normally rewound before listing.

This is equivalent to

LISTTY,I=FILE,O=TTYTTY,S1,L0,C1-72.

Examples:

1. LISTTY,I=TAPE2,10-20,C20-92.

   List columns 20-92 of lines 10-20 of file TAPE2.

2. LISTTY,I=LIST,B,NS,SAVE.

   Save these parameters on ZZZZDFS. Subsequent calls will list from file LIST, suppressing extra blanks and omitting sequence numbers.

3. LISTTY,I=LISTING,7/FORMAT/,P1.

   List all lines on page one of LISTING that contain the word FORMAT beginning in column 7.

4. LISTTY,I=LISTING,COPY,P1-2,ID=/PGM/,O=OUT.

   Copy pages 1 and 2 of program PGM from file LISTING to file OUT, omitting the carriage control character and sequence numbers that are normally generated by LISTTY. If LISTING does not contain a program listing, or some other kind of coded output in paged format, the results will be unpredictable.

2.6.3 Listing Portions of a File at the Terminal, Page-by-Page — PAGE

PAGE lists the contents of a coded file at the terminal page by page. It is most useful for leafing through large files; skipping portions to reach the desired lines or searching for specific character strings.

PAGE subdivides a file into blocks called pages which can be selectively displayed at your terminal. PAGE is designed for use with a CRT terminal which displays output on a TV-like screen. PAGE is called as follows:

PAGE[,optional parameters].

All parameters for PAGE are optional and may appear in any order. Categories include file positioning parameters, search strings and input/output parameters. The following parameters are recognized.

File Positioning:
[dir][pg].[ln]|GLn

dir — direction.
- \+ go forward. This is the default.
- G go to the indicated page.
- \- go backward.

pg — number of pages to move. If directed to the indicated page, pg=page number. If pg=0, the page that was just listed will be displayed again. If pg is not specified, pg=1 unless a period is specified, in which case the default is zero.

A period separates the page number from the line number. If a period is specified, PAGE will position the file at the next page boundary and then move the number of pages specified. If the period is not specified, and no line number is provided, PAGE will not do any extra positioning, and keep the same fractional page offset that may exist.

ln — is the optional page offset line number. A line is what will list in one row at your terminal. This parameter allows a page to be listed from the middle of a page block.

GLn — go to line n. This positions the file at line n and lists one page. Here, a line is defined to be everything between two end-of-line characters on the input file.

Search Strings:
[*|G]{/cstring/|/cstringa/{&|+|$nn}/cstringb|}[c1,c2][U][N]

/cstring/ — a search string up to 199 characters long. Specifying cstring alone causes a listing of all occurrences of cstring on the file.

If two character strings and an operator are specified the following action takes place:

- & true if both strings occur.
- + true if either string occurs.
- $nn true if stringa is followed by stringb with nn characters in between. If nn is omitted, any number greater than or equal to zero will satisfy.

c1,c2 — specifies column location. The character string specified by cstring must begin within the specified range. The default is the same as the column range for listing the file.

U — only search strings which appear as a unit will match. The search string must be bounded by non-alphanumeric characters.

N — The no-match option. If N is not specified, there will be a match when a line contains the string in the column location and format specified by the optional parameters [c1,c2] and U. If N is specified, a match is made with strings which do not contain string in the indicated format.

* — returns PAGE to search string mode using the last search string parameters specified.

Search strings may be continued on consecutive lines if necessary. A search string that is continued will have all trailing blanks removed.

The parameters (c1,c2), U and N may be specified in any order as long as they follow cstring.

G — positions the file at an occurrence of the search string. If G appears alone the last search string parameters will be used.

Input/Output:

I=inlfn — coded input file. This must be a disk file. The interactive default is OUTPUT. The input file is always disconnected. It is rewound before and after processing.

PL=plen — page length, in number of lines per page which will be displayed at the terminal at one time. The interactive default is 20 lines. If the PROMPT option is on, PLEN-1 lines are listed on a page because one line is reserved for the prompt.

W=wid — page width. The number of characters PAGE will output wid before continuing the line. The interactive default is 70 characters. A continued line is indented four spaces.

L=len	line length. This is the maximum number of characters to be taken from an input line. If L=W, lines from the file will not be continued. The form W=L is equivalent. The interactive default is 70 characters.
C_n[m-n]	column range to be listed from the input file. List lines starting at column m up to and including column n. If only one number is specified after C, that will be the starting column. The ending column will be the starting column plus len-1. If two numbers are specified, there must be at least one hyphen between them. The interactive default is 1-70.
O=listlfn	output file. The interactive default is ZZZZOT. If keyword O appears without an equal sign [=] after it, O=ZZZZOT is assumed. ZZZZOT is always connected. If a particular file is specified, it is your responsibility to connect it.
D=dirlfn	directive file name. The interactive default is the connected file ZZZZIN. If a particular file is specified, it is your responsibility to connect it.
LIMITn	page or line limit, where n is a number preceded or followed by a P or an L. The P declares a page limit and the L a line limit.
G*	causes PAGE to restructure the page table up to the current line number.
END	This will terminate PAGE execution after executing any other directives on the same line. PAGE will also end if an end-of-section is encountered on the directive file.
[(nb,]ne)	line or page range in which nb specifies where to begin and ne specifies where to end. Numbers are assumed to be line numbers unless preceded or followed by a P. If only one number is specified, it is used as a stopping point. Your current position is the default starting point.

The following parameters can be ON or OFF. If the parameter appears alone, it reverses the previous setting of the parameter.

SEQ	the presence or absence of sequence numbers. If ON, the listing will be sequenced with line numbers relative to the beginning of the file. The default is OFF.
B	eliminate extra blanks, if more than two consecutive blanks are found. The default is OFF.
Z	to not list end-of-section or end-of-partition messages if ON. The default is OFF.
PROMPT	if ON, PAGE will prompt you with the page number, and the line within the page of the first line listed. If the previous command did not cause PAGE to display another page, you will be prompted with an asterisk. The default is ON.
TEXT	if ON, PAGE will break lines at word borders for continuation unless it requires backing up more than ten characters. The default is ON.

CC — process carriage controls as follows:
1 new page
+ overprint
0 double space
the default is OFF.

ROLL — if ON, a command of +0.n, where n is less than the specified page length, will cause a display of n lines only instead of the complete page. This option should be OFF if the terminal does not have a scroll feature. The default is ON.

The interactive default parameters:
PAGE, I=OUTPUT, O=ZZZZOT, D=ZZZZIN, PL20, W70, L70, C1-70, TEXT=ON, ROLL=ON, PROMPT=ON, CC=OFF.
LIMIT=131070P, LIMIT=131070L, SEQ-OFF, B=OFF, Z=ON, TEXT=ON, ROLL=ON, PROMPT=CN, CC=OFF.

Examples:

1. Suppose you specified:

   PAGE.

   The input file used will be OUTPUT. The file containing directives will be ZZZZIN. The number of lines displayed as page is set to 20; the page width and line length is set to 70. The output file is ZZZZOT. No sequence numbers will be listed, extra blanks will be retained, as will EOS (end-of-section) and EOP (end-of-partition) marks. Also, you will be prompted with the page and line numbers of the first line listed. If your terminal does not conform to these specifications you should specify the appropriate parameters on the PAGE statement.

   You may then select portions of your file to be viewed, page by page.

2. PAGE,I=MYFILE,G/DOG/.

   PAGE positions the file, MYFILE, at the first occurrence of the search string, DOG.

3. In this example, the user is working with a Decwriter. So, he changes the defaults to better fit his terminal.

   PAGE,W=130,L=W,C1=130,I=STATFILE.

Summary and Precedence of Directives:

Any parameter on the PAGE control statement can also be given as a directive to PAGE and vice-versa. A directive line consists of four character fields: keywords, numbers, file names and special characters. Delimiters include blanks, equal signs and commas. If a two number column range is specified, the numbers must be separated by at least one hyphen.

You can put more than one directive on a line. PAGE will process the directives in the following order:

Immediate processing

PL	page length
l	length of line from the file
W	width of the screen
l=W	line length
W=l	line length
LIMIT	line or page limit
Cn-m	column range
B	blank suppression
Z	EOS and EOR message deletion
SEQ	line sequencing
TEXT	line continuation
PROMPT	prompt message/character
CC	carriage control option

If an error is encountered in a line, all of the above commands previously processed will remain in effect. The following commands are processed after the entire line of commands has been decoded. Processing occurs in the following order:

Highest precedence

I=inlfn	input file
O=outlfn	output file
D=dirlfn	directive file
G*	regenerate page table

Next highest precedence

GLn	go to line n
(nb,ne)	range of lines to search

Highest before a string search

+	go forward
-	go backward
G	to this page

String searches

/cstring/	character search string
G	operator
*	return to previous search string parameters.

After a string search

+	go forward
-	go backward
G	to this page

The last thing processed is

END terminate execution

Example:

```
PAGE,I=DATALIST.
+1
G4
/cat/[1-10]
```

PAGE lists the first page of DATALIST; goes to the fourth page, then lists occurrences of the search string CAT (in columns 1-10) from page 4 to the end of the file.

2.7 Manipulating the Authorization File — AUTHORF

The Problem Number Manager uses the utility AUTHORF to change Authorization File (or PN) limits, add and delete user IDs, manage dollar balances, and display information about the problem number account and those of individual users under that PN. In addition, AUTHORF allows individual users to display information about their accounts and to change their passwords.

Full documentation for the AUTHORF utility is available in the *SCOPE/HUSTLER Reference Manual,* Chapter 2. This discussion is dedicated to examples of interactive use of AUTHORF.

After an AUTHORF command is issued, AUTHORF requests input by printing the following prompts:

ADD?	requests input for the ADD directive.
CHG?	requests input for the CHANGE directive.
CMD?	requests an AUTHORF directive.
DEL?	requests input for the DELETE directive.
+?	requests more information for the last directive entered.

2.7.1 Use of AUTHORF by a PN Manager

PN managers control the resources available to a problem number account within the limits set by the Computer Laboratory.

Adding IDs

In the example below, the PN manager is adding IDs to the PN.

```
OK-authorf.

AUTHORF CALLED BY Problem number/user id ON mm/dd/yy
CMD? add,uname id pw dbel
ADD? James cohn 25
ADD? emily nelson 15
ADD? eigen trout 30
ADD? end
CMD? end
```

Changing Limits

In the next example, the PN manager wishes to change the dollar balances for some users, and the PN limit CM for all users.

```
OK-AUTHORF,
CMD? CHANGE,
CHG? id=James,dbal=dbal+10
CHG? id=alden,dbal=25
CHG? id=emily,dbal=dbal+20
CHG? cm=60000
CHG? end
CMD? end
```

change.
id=James, dbal = dbal + 10
id = alden, dbal = 25

cm=60000
end
end

Notice that the CM entry does not have an ID associated with it. Since PN limits apply equally to all users under one account, any changes to PN limits are made for all users. Dollar balances for all users of the PN can be changed by using the keyword ALL, e.g.,

AUTHORF,CHANGE,DBAL=DBAL+50,IDS=ALL.

Deleting IDs

To delete user IDs, the following job could be run:

```
OK-AUTHORF,
CMD? DELETE,
DEL? James
DEL? alden
DEL? end
CMD? end
```

If you wish to delete all users of a PN, use the keyword ALL. This will delete all IDs except the master ID (PN manager's ID).

OK-AUTHORF,DELETE IDS=ALL.

Using VETO

By specifying 'AUTHORF,VETO', you are able to decide whether to keep the additions, changes, or deletions that you have typed. After each line is typed, the change is echoed and followed by a question mark. Respond by typing one of the following:

YES or Y	accepts the change
NO or N	rejects the change
Stop	rejects the change and stops processing the command.
CONTINUE or C	accepts the change and turns off VETO for subsequent changes, thus accepting any other changes.
LIST or L	accepts the change, turns off VETO, and turns on LIST for subsequent changes, processed by the current command.

Using LIST

The LIST option can also be specified by typing 'AUTHORF,LIST'. This will echo each change without allowing a VETO.

Displaying Limits

Both the problem number manager and individual user can display certain information about their accounts. In addition, the problem number manager can display information about individual accounts under the problem number.

The command

AUTHORF,DISPLAY ALL.

will display all of the information in the Authorization File that the user has access to. The command

AUTHORF,DISPLAY LIMITS.

displays the user's PN and job limits.

2.7.2 Changing the Password Associated with your User ID

It is a good idea to change your password regularly in order to protect the funds and information available to your account. Following are three frequently used methods of changing the password using AUTHORF.

1. Type 'AUTHORF,CHANGE,PW=password.' where 'password' is your new password.

```
OK-authorf,change,pw=cloth.
WAITING FOR AUTHORIZATION FILE

AUTHORF CALLED BY 11600/ELAINE      ON 03/17/80
OK-
```

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

```
OK-authorf,change pw.

AUTHORF CALLED BY 11600/ELAINE      ON 03/17/80

ENTER NEW PASSWORD--
          
OK-
```

3. Type 'AUTHORF,CHANGE PW,VETO.' This method allows you 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. Type Y (Yes) or N (No), indicating whether the password is correct. AUTHORF then blacks out the echoed password and, if you entered N, prompts for a new password. If you entered Y, the password is accepted.

```
OK-authorf,change pw,veto,
AUTHORF CALLED BY 11600/ELAINE      ON 03/17/80

ENTER NEW PASSWORD--
■■■■■■■■■■■
THYME ?■
```

Note: In method 3 V may be typed instead of VETO.

2.8 Using Paper Tapes and Magnetic Tape Cassettes

Five system commands perform input/output operations with paper tape or magnetic tape cassette: TAPE, TAPEC, TPREAD, READPT and WRITEPT. TAPE and TAPEC read from tapes into EWFILE, instructing EDITOR to enter the lines in the same format as lines entered manually. EDITOR accepts only text lines under TAPE, but processes both text lines and EDITOR directives under TAPEC. Both TAPE and TAPEC are discussed in Chapter 3. Unlike TAPE and TAPEC, READPT and TPREAD do not modify an EDITOR workfile; they copy the contents of a tape to a disk file. READPT and TPREAD are identical, except that TPREAD can automatically start and stop the tape reader.

Note: Tapes may be read reliably only from terminals equipped with a tape reader which responds to the control characters (ASCII code - DC1) and (ASCII code - DC3); Reader - ON and Reader - OFF.

Each line entered from tape must be terminated by a carriage return.

After receiving a tape command, the message 'READY FOR TAPE' is printed at the terminal; you then start the tape through the tape reader at the terminal. After the tape has been read, you must indicate end-of-tape by entering the abort character[1]. After an escape, wait for the EOT message before sending more data. This is especially important when using minicomputers and "smart" terminals that transmit volumes of data at high speeds.

2.8.1 Copying the Contents of a Paper Tape

The READPT and TPREAD commands write unnumbered lines read from tape or cassette to the file designated by lfn in the format specified by cc.

READPT,lfn[,NR][,cc]
TPREAD,lfn[,NR][,cc]

lfn the name of the file onto which the contents of the tape are written. The file is rewound both before and after the copy.

NR no rewind, the file will not be rewound before or after the copy.

[1]The default abort character is the escape key (ESC). See Section 8.1.1.

cc character code used on the tape. Valid options are:

DC	Display Code
AS	ASCII
AF	ASCII Fancy
BI	Binary
BF	Binary Fancy

If cc is not specified, DC is assumed. Lines longer than 240 characters are automatically broken. A new line would begin with the 241st character.

Indicate the end-of-tape only by depressing the escape key, ESC, unless you are reading a binary file which requires the use of the break key to halt input. After an escape, wait for the EOT message before sending more data. This is especially important when using minicomputers and "smart" terminals that transmit volumes of data at high speeds.

For instructions on performing I/O on binary files, see Appendix C.

TPREAD automatically starts and stops the tape reader. READPT does not. READPT should be used for slow (110 baud) data transmission or in conjunction with %READER,ON. See Section 8.3.13.

2.8.2 Copying the Contents of a Local File to a Paper Tape or Cassette — WRITEPT

The WRITEPT command writes the contents of a local file at the terminal.

WRITEPT,lfn[,NR][,cc].

lfn the name of the local file which contains the information to be written on paper tape or cassette. lfn is disconnected when the operation is complete.

NR no rewind, the file will not be rewound before or after the copy.

cc the character code used in the file. Valid options are:

DC	Display Code; the default character set.
AS	ASCII
AF	ASCII Fancy
BI	Binary; for instructions on performing output on binary files, see Appendix C.
BF	Binary Fancy

2.9 Receiving Information at the Terminal

This section describes the on-line documentation facility on the MSU computer system, and other utilities for retrieving information about job and system resources.

2.9.1
Using On-line Documentation — HELP

The HELP file is an on-line general reference source for all users. The entries are written in reference manual style and should serve as a reminder rather than a learning aid.

Information is stored in the HELP file under entry names. The entries are of two types. The first type gives detailed information about the use of a particular product or command. Some of the entries of this type (and examples of entry names) are:

Control Statements - PFLIST,LISTTY
EDITOR Directives - SAVE,SYSTEM
Front-End Commands - LOGIN,JOBSTAT
Plotting Packages and Routines - GCS, PLOT
Programming Packages - SPSS,APEX

The second type of HELP entry offers general information and news. Some of the entries of this type are:

Computer Laboratory Services - HOURS, SCHEDULE, HOLISCHED
Product Information - EDITOR, FRONTEND
Computer Laboratory Announcements - NEWS

The HELP entry has three parts: title, abstract and body.

The **title** is a one line description of the item's function.

The **abstract** is a general discussion of the item's function.

The body contains the calling sequence, followed by a parameter list. This will contain default values and any differences between interactive and batch processing. Cross references to other Computer Laboratory or CDC documentation appears at the end of the body.

You can specify which portions of a HELP description to be displayed with the retrieval parameters on the HELP control statement.

HELP[,optional parameters].

There are two types of parameters on the HELP control statement: file parameters and retrieval parameters.

The two most important file parameters are O=listlfn and L*usrlib. If O=listlfn is specified the HELP description is output to the file listlfn rather than displayed at the terminal.

If L*usrlib is supplied as the first HELP parameter, the given user library, rather than the HAL main library, is searched for the desired HELP description(s).

Retrieval parameters indicate which HELP descriptions are desired. Up to 30 may be specified in one call to HELP. Each parameter consists of a library entry name or keyword, optionally prefixed by various modifiers. Some of the retrieval forms are discussed below:

Specifying the descriptions to be printed:

You may specify one or several HELP descriptions by using the appropriate keyword.

keywrd where keywrd can be any of the following:

a legal entry name
A category name
ALL (for every entry on the library) or several other options discussed in the *HAL Reference Manual* and HELP,F*HELP

Specifying which part of the descriptions will be printed, using prefixes:

T*keywrd	**list description title(s)**
A*keywrd	**list description abstract(s)**
F*keywrd	**list full description(s)**
keywrd	**list description using segmentation. (Many HELP entries are broken into segments. At the end of each segment, you are asked whether or not you wish the listing to continue. Specifying a keyword with no prefix, allows you to halt a listing or continue at each segment boundary.)**

Examples of interactive use:

1. HELP,F*ALL.
   produces a complete listing of all descriptions on the HUSTLER Auxiliary Library. This is an extremely long, expensive listing.

2. HELP,*HOLISCHED,NEWS,F*BANNER.
   produces a listing containing the title of HOLISCHED, an abstract of NEWS, and a full listing of BANNER.

3. HELP,L*MYLIB,NEWRTN.
   produces an abstract of the description NEWRTN from the user library, MYLIB.

4. HELP.
   displays an introduction to the HELP utility.

5. HELP,PFLOAD.
   interactively this produces a segmented listing of the PFLOAD description, which the user can choose to continue or terminate at the end of each segment.

```
OK-[redacted]

******** PFLOAD
Reloads permanent files from a dump tape.

BRIEF OR LONG?B
PFLOAD allows the user  to reload information from both user and system
dump tapes.
END OF ABSTRACT. DO YOU WANT MORE?N
Calling sequence:

PFLOAD[,MT[=vrn[=...]][,NT[=vrn[=...]][,ALL][,I=lfn[=...]][,PFN=pfn]
       [,CY={XX|ANY|ALL}][,RP=xx][,DUP={IGNORE|NEWNAME}]
       [,O=outlfn][,U=unlfn].
```

```
where categories include tape specification, PF selection, recatalosing
information, output file specification.
MORE?
Tape Specification

MT[=vrn[=...]]          specifies 7-track tapes to be loaded. If MT
                        appears alone, the tapes are specified in the
                        input list.

NT[=vrn[=...]]          specifies 9-track tapes to be loaded. If NT
                        appears alone, the tapes are specified in the
                        input list.

MORE?
PF Selection

ALL                     reload all PFs on specified tapes.

I=lfn[=...]             specifies up to 5 local file names which hold
                        a list of PFs to be reloaded.

PFN=pfn                 specifies a single PF to be reloaded. If pfn
                        contains special characters, delimiters must
                        be used.

CY={xx|ANY|ALL}         specifies cycle to be reloaded.

                        where:

                        xx      is the cycle number.
                        ANY     causes first cycle encountered to be
                                loaded.
                        ALL     causes all cycles of the pfn to be
                                loaded.

MORE?
OK-
```

6. HELP,L*UNSUP,F*UNSUP,F*PFM.
displays two entries on the Computer Laboratory Unsupported library: UNSUP, which includes a list of all utilities on L*UNSUP and PFM, a description of a group of permanent file management utilities.

2.9.2
Displaying Job Status — STATUS

STATUS prints information about the status of any job on the system.

HAL,STATUS[,joblist][,REPEAT=n].

joblist — a list of one or more job sequence numbers, separated by commas, which specify the jobs for which status information will be printed. If no sequence numbers are given, STATUS will ask for sequence numbers.

Sequence numbers in the joblist can be full sequence numbers or abbreviated forms. If 1-6 characters are given, all jobs that end with those characters will be listed.

Typing 'HAL,STATUS,123' could yield the following result:

TB57123
SA24123
IB10123
MV85123

If any character of a sequence number is replaced by *, that character is ignored in matching.

Typing 'HAL,STATUS,TB141*1' could yield the following result:

TB14131
TB14171
TB14101

REPEAT=n — if you specify the REPEAT parameter, the status of the requested job(s) will be printed approximately every n seconds.

Note: 'STATUS,*' will list all jobs on the system, and can be quite lengthy.

2.9.3
Displaying System Resources — ASSETS

ASSETS causes the system to display the limits and status of certain system resources and the current settings of software mechanisms such as REDUCE and DAYMSG. Below is a sample and explanation of output from ASSETS.

```
FILES:  MAX 35,IN USE 5 TIME LIMIT- 454B RTL-0010B EXIT MODE-7
MAX FL 120000 CURRENT FL 040000 SWITCHES ON- 2 5
CP TIME 8.746 PF TIME 28.153 APPROX $ VALUE 1.19
AUTORFL-ON MAP-OFF REDUCE-ON PROMPT-OFF LOCK-OFF DAYMSG-ON
```

FILES: MAX — the maximum number of files that may be assigned to you at any one time during an interactive job. If exceeded, the system prints FILE LIMIT EXCEEDED and you must return some of your files before you can execute any command other than RETURN or FILES.

IN USE	the number of files, including system files, currently assigned to the terminal.
TIME LIMIT	the maximum octal number of CPU seconds that may be used during this interactive session.
RTL	the current time limit, in octal seconds, allowed for execution of each line of commands.
EXIT MODE	indicates the current halt conditions for execution of central processor programs, as specified by the MODE command. Under the default condition 7, execution aborts if the program attempts to reference an out-of-range address, or an out-of-range (infinite) or indefinite operand.
MAX FL	the maximum field length (octal) that can be requested by an RFL command.
CURRENT FL	the current user field length (octal) as established by an RFL command or default.
SWITCHES ON	a list of all sense switches currently ON as a result of the SWITCH command.
CP TIME	the number of CPU seconds used since you logged in.
PP TIME	the actual number of PP seconds used since you logged in.
APPROX $ VALUE	the approximate cost of the session thus far, at Rate Group 3, excluding the connect time charge.

The last line gives the ON, OFF, or PART condition of various system mechanisms.

2.9.4 Controlling the Display of Dayfile Messages — DAYMSG

DAYMSG[,ON|,PART|,OFF].

The DAYMSG command, intended especially for large UPDATE and compilation runs, allows you to suppress dayfile messages. The options are:

ON allows all dayfile messages to appear at the terminal. This is the default condition.

PART suppresses many messages that would normally appear at your terminal (e.g., "COMPILING name" and "UPDATING deckname"), but does not suppress those messages which normally appear in the dayfiles of batch jobs.

OFF suppresses all dayfile messages, including many important error messages. Use with caution.

2.9.5
Determining Tape Status — LISTAPE

LISTAPE.

The LISTAPE command lists the visual reel names of all tapes that have been requested by the various jobs in the system, but not yet mounted and assigned by the operator. In order to access information stored on magnetic tape, interactive users must DISPOSE, to batch input, a job file that copies the information from tape to a permanent file (see Chapter 7). LISTAPE helps indicate, but does not determine, whether a copy of your tape file is ready for interactive access. NOTE: A tape name will be displayed by LISTAPE only after the batch job has requested the tape while at a control point. In addition, LISTAPE does not display the tape name after the operator has mounted and assigned the tape, even though the batch job may not have completed processing the tape.

3

EDITOR—The Text Editing System

3.1
General Description

Introduction

EDITOR is the SCOPE/HUSTLER text-editing system. It enables you to build and edit files. These files may contain programs, data or text.

EDITOR can operate in two modes:

1. DC (Display Code), which produces output files in the Display code character set. This character set contains 63 characters — upper-case letters, numbers and special characters. This is commonly used for programs, SPSS directives and DISPLAY code data. Data processed by these programs is also represented as Display code.

   In this mode, EDITOR supplies you with a comprehensive tool for program development. The program text can be entered, modified, compiled and corrected. Data files can also be corrected. The program can be executed, tested and run in combination with other programs and exec files (see Section 3.12).

   DC is the default mode of operation for EDITOR. EDITOR will automatically prepare output files in the DC character set unless you specify otherwise.

2. AF (ASCII Fancy), which produces files in the ASCII Fancy character set[1]. This character set includes 128 characters — upper and lower case letters, numbers, special characters and control characters. This can be used for BASIC programs and for text processing operations. To process ASCII files using EDITOR, we recommend specifying "SET,AF=ON." This causes EDITOR to process and output your file using the full ASCII character set. (Additional parameters you should consider when working with ASCII files are CASE and CTRL. See Section 3.21.)

To achieve greater power and flexibility, EDITOR is designed to perform all text manipulations on the contents of a special work file named EWFILE. This chapter deals with directives which enter text into EWFILE, edit it, and then copy it into a standard coded file (see the *SCOPE/HUSTLER Reference Manual*, Chapter 4).

[1]In the ASCII Fancy character set, characters are packed 8bits-in-12, 5 characters per word. An end-of-line is represented by a 60-bit word containing zero in the right most 12 bits, as in SCOPE coded files. A null (ASCII code 00) is represented by 4000_8 in 12 bits. ASCII Fancy is the only ASCII representation EDITOR can handle. NOS 6/12 and other upper/lower character sets are not supported.

Chapter Directory

The early sections of this chapter should give you sufficient information to begin to use EDITOR. As you become more familiar with the use of EDITOR, you can acquire more sophisticated techniques by reading on in the chapter. The following directory outlines the contents of each major section of the chapter and should help you find the information you need.

3.1 General Description
Contains an introduction to EDITOR, the chapter directory, a directive index and a discussion of the basic ideas used in EDITOR.

3.2 Format of Text Lines
Defines an EDITOR text line, line numbering conventions and the use of tabulation.

3.3 Format of EDITOR Directives
Describes the components of an EDITOR directive, the notation used in this chapter and some sample directives.

3.4 EDITOR Parameters and Options
Discusses the common parameters and options which are used with EDITOR directives.

3.5 Editing Systems
Describes the standard formatting systems.

3.6 Text Line Formatting Directives
Discusses specific formatting directives such as line length, left margin, tab stops and tab characters.

3.7 Getting Lines into EWFILE
Discusses different methods for entering text lines into an EDITOR work file.

3.8 Listing the Contents of EWFILE at the Terminal—LIST
Discusses how to list all or part of an EDITOR work file at the terminal. Describes methods for storing an EDITOR work file on another file; in work file format, standard coded format or in a form suitable for punching as a paper tape.

3.9 Cataloging/Scratching and Use of Alternate EWFILES
Contains instructions on cataloging EWFILE as a permanent file, getting rid of the contents of EWFILE and using a previously created EWFILE.

3.10 Compilation Directives
Describes how to compile and execute the contents of an EDITOR work file.

3.11 Disposing a Job to Batch—BATCH
Describes how to use EDITOR to create and execute a batch job.

3.12 Exec Files and EDITOR—GO
Discusses the use of exec files with EDITOR. An exec file contains a sequence of SCOPE/HUSTLER commands.

3.13 Inter-line Editing
Discusses ways to alter an EDITOR work file on a line-by-line basis; by moving, duplicating, deleting or resequencing text lines.

3.14 Intra-line Editing
Describes ways to alter text lines within an EWFILE by altering the contents or structure of text lines. Characters can be replaced or inserted and lines can be truncated or continued beyond the margin.

3.15 EWFILE Segmentation
Discusses how to divide the EWFILE into segments which can be formatted differently.

3.16 Abbreviations for Character Strings
Shows how you can create your abbreviations for frequently used combinations of parameters.

3.17 EDITOR Work File Status—EDSTAT
Describes the EDSTAT command which allows you to display the current EDITOR work file attributes.

3.18 Locking the Work File—EWFLOCK
Shows how to protect EWFILE from accidental alteration.

3.19 Changing Default Conditions—SET
Shows how to alter the default setting for EDITOR directive parameters.

3.20 UPDATE and EDITOR
Describes how EDITOR can be used to create correction sets for an UPDATE program library.

3.21 Using EDITOR to Process ASCII Fancy Files
Describes how EDITOR can be used to process files containing full ASCII data (i.e. both upper and lower case).

Directive Index

EDSTAT	displays current EDITOR work file attributes.	3.17
EOT	terminates processing of paper tape input.	3.7.4
EWFLOCK	protects EWFILE from accidental alteration.	3.18
FOLD	truncates all lines longer than the current line length. Continuation line processing depends on the editing system in force.	3.14.4
FORMAT	defines a line format and defines the boundaries of that format within the EDITOR work file.	3.15.1
FTN	compiles a FORTRAN program with error checking.	3.10.4
FTNER	compiles and executes a FORTRAN program, using error checking.	3.10.4
FTNX	compiles and executes a FORTRAN program with no error checking.	3.10.4
GO	saves specified text lines and executes commands from a specified exec file.	3.12
INSERT	inserts contents of a specified file at a certain point in the EDITOR work file.	3.13.5
LENGTH	sets maximum line length.	3.6.1
LIST	lists specified lines at the terminal.	3.8.1
LISTF	lists text lines onto a file in EDITOR work file format.	3.8.3
MARGIN	sets the left margin.	3.6.2
MERGE	merges the contents of a file into an EDITOR work file in order by line number.	3.13.6
MOVE	moves specified text lines to a new location.	3.13.1
N	initiates automatic line numbering.	3.7.1
OLD	enters the contents of a file into an empty EDITOR work file.	3.7.3
READ	enters text lines and executes EDITOR directives contained on a file.	3.7.2
RESEQ	renumbers text lines without altering their order.	3.13.4
SAVE	copies text lines from an EDITOR work file to a file, converting each text line to a SCOPE unit record (or "card image").	3.8.2
SCRATCH	returns the current EDITOR work file and creates a new one.	3.9.2
SET	changes the default setting for EDITOR options.	3.19
STRING	allows the user to define an abbreviation which can be used instead of a larger string of characters.	3.16.1

SYSTEM	specifies the editing system which formats text lines.	3.5
TAB	specifies up to seven tab stops.	3.6.3
TABCH	defines a tab character.	3.6.4
TAPE	transmits text lines from paper tape.	3.7.4
TAPEC	transmits text lines and directives from paper tape.	3.7.4
USE	causes a file in work file format to become the current EDITOR work file.	3.9.3

Basic Ideas

In batch computing, you prepare a card deck with the keypunch, proof the deck by making a listing on the lister-printer and submit the deck by reading it into the computer via the card reader. The interactive system allows you to prepare control statements, programs and data interactively. The power and convenience of the interactive system makes this an attractive option. You can perform all phases of job preparation interactively and evaluate the results of each stage immediately.

The MSU Text Editor plays an important role in this process. You can tailor the EDITOR work file to fit your needs. When you issue any EDITOR directive interactively, you are assigned an EDITOR work file named EWFILE, which (like INPUT, OUTPUT, and PUNCH) is a special file name. All EDITOR directives operate on the contents of EWFILE, and because of the special work file format, only EDITOR can process EWFILE. So in order to edit a file—say MYFILE—you must have EDITOR read MYFILE into EWFILE and convert it to the special format. If MYFILE is already in the work file format, it must still be renamed EWFILE by the USE directive. Conversely, you must copy the contents of EWFILE into a standard SCOPE coded file before other programs can use them. To retain a work file for later use, however, you need only catalog EWFILE as a permanent file (see Section 3.9.1).

EWFILE is composed of text lines, and each is associated with a unique line number that determines its position within EWFILE. EDITOR always sorts text lines into ascending order according to their respective line numbers.

You can enter text lines into EWFILE from either the terminal or another file. Each text line entered from the terminal, whether typed directly or read from paper tape or magnetic tape cassette, **must begin with a line number**, whereas text lines entered from a file need not contain line numbers. EDITOR will assign line numbers automatically as the file is converted to work file format.

As text lines are entered into EWFILE from the terminal, they are formatted according to a set of work file attributes. These include a maximum line length, a left margin, tab stops, tab character and an editing system which can automatically format lines for BASIC, COMPASS or FORTRAN programs. (Note that the default editing system is FORTRAN.)

The remaining EDITOR directives manipulate text lines within the work file. These enable you to:

1. delete lines or line ranges from the work file,

2. insert or merge lines into the work file from another file,

3. move lines from one portion of EWFILE to another,

4. duplicate lines in other portions of EWFILE,

5. list all or selected lines of the work file,

6. scan, and perform EDITOR operations on lines containing a specifiable character string,

7. scan, and perform EDITOR operations on selected lines or line ranges of EWFILE,

8. replace, delete, and insert character strings within selected text lines.

Warning

Avoid using the abort character (default=ESC) to abort execution of EDITOR directives. If a user abort is signalled there may be cases where, despite software safeguards, lost information is unavoidable.

There is a safer way to halt execution of EDITOR directives. All EDITOR directives provide a VETO option (described in Section 3.4.5) which allows you to check the result of an operation before each text line is modified, and to terminate execution safely if an error has been made.

3.2
Format of Text Lines

A text line input to EDITOR has three components: a line number, text characters and tab characters.

A line number is required. You may assign line numbers on a line-by-line basis or use the auto line numbering facility to speed text entry (see Section 3.7.1).

At least one text character (or blank) is required after the line number. Entering a line number alone will delete a line with that number if one exists or do nothing at all if no line with that number exists.

Tab characters are optional. They allow you to format your entries using up to seven tab stops. Both tab characters and tab stops may be set by the user or by default. This facilitates text entry, for both programs and data (see Section 3.6.4).

3.2.1
Line Numbers

Line numbers determine the position of text lines within the work file. If a new text line is entered which has the same line number as an existing text line, the new line replaces the old line. If a line is entered which consists only of a line number, any existing text line having the same line number is deleted. If no such line exists, EDITOR ignores the entry, i.e., it does not create a blank text line. In order to create a blank line, at least one blank must be typed following the line number. For example, suppose text lines are entered as follows:

```
100PROGRAM EXAMPLE (OUTPUT,TAPE1)
110INTEGER DATA
120DIMENSION DATA (20)
130INTEGER DATA (20)
120
130K=0
140K
```

This is equivalent to:

```
100=PROGRAM EXAMPLE(OUTPUT,TAPE1)
110=INTEGER DATA(20)
130=K=0
140=1
```

3.2.2 Line Number Formation Rules

Line numbers are of the form:

nnnnnn.mmmmmm

where nnnnnn is an integer and mmmmmm is a decimal fraction. The following rules apply to the formation of line numbers:

1. The integer part may consist of 1-6 digits. Thus, .mmmmmm is illegal, while the form 0.mmmmmm is legal.

2. The decimal fraction may consist of 0-6 digits. Thus, line numbers may not exceed 999999.999999.

3. When a line number is used in an EDITOR directive and there is no decimal fraction, the decimal point must be omitted. For example, 100 and 100.0 are correct, but 100. is incorrect.

4. Line numbers prefacing a text line are terminated either by

   a. a blank or non-numeric character which is a part of the text line, or

   b. an '=' (equal sign) which is not treated as a text character.

   Unless special reformatting occurs under the editing system (see Section 3.5), column 1 of the text line begins with the first character which follows either the line number in case (a) or the equal sign in case (b). Thus, the terminating equal sign allows you to enter a text line which begins with a digit, keeping the digit separate from the EDITOR line number.

Examples:

```
100Z COLUMN 1 CONTAINS A Z.

200=Z COLUMN 1 CONTAINS A Z.
```

Both entries produce the text line:

```
Z COLUMN 1 CONTAINS A Z.
```

However, under SYSTEM GENERAL (Section 3.5.5) or SYSTEM TEXT (Section 3.5.6):

```
100=3 WHAT IS IN COLUMN 1?

200 3 WHAT IS IN COLUMN 1?
```

produce different results:

```
3 WHAT IS IN COLUMN 1?      (line 100)

 3 WHAT IS IN COLUMN 1?     (line 200)
```

3.2.3 Using the Tab Character

Tabulation is the systematic arrangement of data in rows and columns for ready reference. This is useful for both program text and data. Material which is arranged in logical blocks is more readily understood and corrected.

The use of a tab character simplifies the use of tabulation. You can move to the desired column using the tab character without counting individual spaces. This saves time and improves accuracy.

You can choose a tab character with the TABCH directive (see Section 3.6.4). There are two criteria for this choice:

1. The tab character should be easy to use. Be sure the character is in a convenient location on the terminal keyboard.

2. Try to pick a character which would not normally appear in the text. For example, if backslashes are used in the text, the backslash would be a poor choice for a tab character.

The EDITOR tab character does not act in the same way as the terminal TAB key. When you enter a tab character, the carriage does not move to the next tab position. Instead, EDITOR replaces the tab character with enough blanks so that the next character will fall in the next tab position.

You may set 'TABCH' equal to the control character 'Control-I' which will allow you to use the tab key for tabs and cause the carriage to move. Type 'TABCH' followed by a comma, depress the CTRL key and hit the 'I' key followed by a carriage return. This assumes, however, that your terminal supports this function and that tab columns have been properly preset.

The default tab character is the semicolon.

The tab character is recognized and interpreted in text typed at the terminal and text entered by the READ command, but not in text entered by OLD, INSERT or MERGE.

3.3 Format of EDITOR Directives

Each EDITOR directive consists of a directive verb optionally followed by one or more parameters. The directive verb must be separated from the first parameter by a comma or a blank. Additional parameters must likewise be separated from each other by commas or blanks. Parameters should appear in the order presented in the following descriptions unless otherwise specified.

In this document, directive verbs always appear in capital letters. When parameters appear in capital letters, they represent key words to EDITOR; otherwise, they appear in lower case and represent values to be supplied by the user. Brackets indicate that the enclosed parameter is optional. If two directives appear on the same line, they must be separated by a period.

Examples:

OLD lfn [FROM n].

In the syntax of the directive, OLD is the directive verb and 'lfn' is a required parameter, while remaining parameters are optional. Both lfn and n (if used) must be replaced by user-specified values. Note that 'FROM n' is treated as a single parameter, and therefore FROM and n must be entered in the order shown. To further clarify the notation of this chapter, valid forms of the directive OLD are shown below.

OLD MYFILE,FROM,100.

OLD MYFILE

OLD,MYFILE FROM 100

OLD MYFILE FROM, 100

OLD MYFILE, FROM 100

Similarly, the following entries would be invalid, for the reasons given.

OLD,MYFILE,100,FROM.	FROM must precede the value, 100.
OLD.	According to the syntax, the directive verb, OLD, must be followed by a file name.
OLD,MYFILE LIST.	A period must follow MYFILE to separate the directives OLD and LIST.

3.4 EDITOR Parameters and Options

The following is a list of the most common parameters and options used with EDITOR directives. This serves as a general introduction to the parameters. The use of parameters with specific directives is discussed with the directives. A complete list of all parameters appears in Appendix G.

3.4.1 lnum

The parameter lnum can define a set of up to twenty line numbers, or line number ranges. In this context, a line number can be any of the following:

1. An integer or decimal number as described in Section 3.2.2. For example:

   100

   0.01

   100.01

   123456.123456

2. A line number range is of the form, n-m, where n and m are both line numbers as defined by the rules in Section 3.2.2, and where the value of n is less than that of m. An inclusive line number range, n-m refers to all text lines between, and including, those represented by n and m.

   Exclusive line ranges may be specified by adding an X suffix to n or m or both n and m. This refers to all text lines between n and m but excluding, those represented by nX or mX. The line number associated with the X suffix is excluded. All of the following are valid line number ranges:

   100-201.52

   100X-123456.654321

   1-99X,100,200,100-2000,3000X-4000

   100X-200X

3. One of the following special symbols:

   * representing the line most recently processed by the last EDITOR directive or the last line entered as a text line. The last line may have been referenced using the VETO option. Even if you vetoed the directive, * indicates that last line. You must be sure that * refers to the intended line.

   *F representing the first line of the work file.

   *L representing the last line of the work file.

   *A representing all lines in the work file. This is equivalent to *F-*L.

   All of the following are valid line ranges:

   *-*L

   *F-150

   *X-*L

4. A line number followed by a line count, and having the form m+n, where m is a line number as defined in (1) or (2) and n is a 1-6 digit non-zero integer. This form denotes the text line n lines past the text line indicated by m. For example, if EWFILE consists of four text lines numbered:

   100, 100.01, 100.3 101, then

100+2	refers to	100.3
100+3	refers to	101
100.3+1	refers to	101

5. A line count alone, having the form, +n, where n is a 1-6 digit non-zero integer. This form denotes the text line n lines past that indicated by the previous line number in the lnum parameter (a positive line offset). If +n is the first line mentioned, EDITOR substitutes 0+n. Thus, the parameter:

   300,+2,+3

   refers to three lines: line 300, the line two lines past 300, and the line five lines past 300. The parameter:

   100-+25

   refers to the line range beginning at line 100 and ending 25 lines beyond line 100.

6. A line number followed by a line count and having the form m\n, where m is the line number as defined in (1) or (2) and n is a 1-6 digit non-zero integer. This form denotes the text line n lines before the text line m (a negative line offset). For example, *L\5 refers to the text line 5 lines before the last text line in EWFILE.

7. A line count alone, having the form \n, where n is a 1-6 digit non-zero integer. This form denotes the text line n lines before the previous line number (e.g. 300,\2 refers to line 300 and the line 2 lines before 300).

To illustrate the use of the lnum parameter, suppose text lines are entered into an empty EWFILE by typing:

```
1A
2B
3D
0.1THIS IS A TEST
4E
2.5C
```

In the examples below, the lnum parameter specifies the text lines to be listed from the EWFILE just created.

1. LIST, *.

   ```
   2.5=C
   ```

   Here, the last text line processed is the last text line entered.

2. LIST 2-3.

   ```
   2=B
   2.5=C
   3=D
   ```

   Note that the range, 2-3, consists of three lines because line 2.5 was inserted between lines 2 and 3.

3.4.2 c

The c parameter represents a column number. (c must be an integer value.) It can be used in two ways:

1. To specify individual columns;

   c or $c_1,c_2,...c_i$

   In this form, c represents a single column or a series of columns. It is used in the LENGTH and MARGIN directives in single column form. When used with the TAB directives it may be used to specify a series of columns.

2. To specify a column range;

   (c_1,c_2)

   This form describes a column range. The directive processes data in all columns from c_1 to c_2. This is used in intra-line editing.

3.4.3 txt

Simple Character Strings

The txt parameter defines a character string, which is used to specify which lines of EWFILE are to be processed by the EDITOR directive in question. The txt parameter has the form:

/chars/[(c_1[,c_2])][U][N][C]

The parameter elements (c_1,c_2), U, N, and C, may be specified in any order as long as they follow /chars/. None of the four elements should be separated from the others by blanks or commas.

If N is not specified, the EDITOR directive is performed only on lines which contain the character string /chars/ in the column location and format specified by the optional subparameters (c_1,c_2) and U. If N is specified, the directive is performed only on lines that do not contain /chars/ in the indicated format.

NOTE: The interpretation of /chars/[(c_1[,c_2])][U]N is the opposite of /chars/[(c_1[,c_2])][U]. Therefore, references to txt in the remainder of this chapter will assume that N is omitted. In other words, /chars/[(c_1[,c_2])][U] is taken to be the standard txt format.

/chars/ a character string of up to 140 characters delimited by slash marks (/). A slash within a string is represented by two consecutive slash marks, e.g., the string A/B is represented by /A//B/.

(c_1,c_2) a column range as described in 3.4.2. If specified, EDITOR directives are performed only on text lines in which the specified character string begins within the column range denoted by (c_1,c_2), or begins in the column specified by (c_1).

```
LIST,/READ/(7,9).
```

This directive lists all text lines in which the string, READ, begins in columns 7, 8, or 9.

```
LIST,/READ/(7)
```

This directive lists all text lines in which the string, READ, begins in column 7.

U — If specified, EDITOR directives are performed only on text lines in which the specified character string occurs as a unit, i.e., it must be bounded by non-alphanumeric characters. (Non-alphanumeric characters are those whose octal Display code values exceed 44 octal, e.g., blanks, commas, periods). The U modifier affects only the text string to which it is joined. This differs from the UNIT parameter which affects all text strings in the directive.

N — The 'no match' modifier specifies that the EDITOR directive is performed only on lines that do not contain the character string specified by /chars/[(c_1[,c_2])][U].

C — the 'case' modifier specifies that the case (upper/lower) of the characters in the search string is to be considered in locating a match. Normally case is not considered when searching for a match because both the string and the line to be searched are folded to upper case before searching. This modifier is only useful when processing ASCII files. It is ignored if AF has not been specified.

Compound Character Strings

The txt parameter can be used to define compound text search strings. There are four different types of conjunctions.

1. $txt_a + txt_b$
2. $txt_a \& txt_b$
3. $txt_a \$ txt_b$
4. $txt_a \$ nn txt_b$

These have the following effects on lines to be processed by EDITOR.

1. All lines in which either txt_a or txt_b is satisfied will be processed.

2. All lines in which both txt_a and txt_b are satisfied will be processed.

3. All lines in which both txt_a and txt_b are satisfied with txt_bfollowing txt_a and any number of characters between the two will be processed.

4. All lines in which both txt_a and txt_b are satisfied with txt_bfollowing txt_a by exactly nn characters will be processed.

A compound character string can be used in any statement that allows the txt parameter, except for the txt_1 and txt_2 parameters in the intra-line editing directive.

Examples:

Suppose EWFILE contains these FORTRAN statements:

```
100=        DO 5 I=1,10
110=5       DONE=DONE+DID(I)
320=        PRINT 200, SKIDO23
550=200     FORMAT (21HNUMBER OF TASKS TO DO,   I8)
```

The LIST directives below illustrate the different forms of the txt parameter.

1. LIST,/DO/.

   All four text lines would be listed.

2. LIST,/DO/U.

   The string, DO, occurs as a unit in lines 100 and 550.

3. LIST,/DO/(7).

   The string, DO, begins in column 7 in lines 100 and 110.

4. LIST,/DO/(7)U.

   Only line 100 contains DO in the format specified.

5. LIST,/DO/(7)UN.

   Lines 110, 320, and 550 are listed.

6. LIST,/DO/$3/I/

   Lines 100 and 550 are listed.

3.4.4 UNIT/NUNIT

The UNIT parameter is used in conjunction with the txt parameter. It specifies whether or not the character string must appear as a unit within the text line. A character string is considered a unit if it is bounded by non-alphanumeric characters. The UNIT parameter affects all text strings in the directive. It can be used with the following directives: DELETE, DUP, LIST, LISTF, MOVE, SAVE, SET and intra-line editing.

UNIT may be abbreviated U. NUNIT may be abbreviated NU. The default is NUNIT. This may be altered using the SET directive.

3.4.5 VETO/NVETO

The VETO option allows you to examine each line processed by an EDITOR directive before the operation is performed on the line. You can decide what action to take by typing an action code. Each text line affected by the directive is to be displayed at the terminal before the operation is performed on that line. EDITOR outputs a '?' (question mark) on the next line, and you respond by typing one of the following:

Y (YES)	perform the operation on this line. VETO continues.
N (NO)	do not perform the operation on this line. VETO continues.
A (ACCEPT)	perform the operation on this line, and then terminate the operation.

S (STOP)	stop the operation (without editing the current line).
C (CONTINUE)	perform the operation on this line and continue processing remaining lines without VETO or LIST.
L (LIST)	perform the operation on this line and continue processing remaining lines without VETO but list all processed lines in full.
K (KILL)	stop the operation (without editing the current line) then exit EDITOR via a CPU abort.
Yn	perform the operation without VETO on the next n lines (including the current line), where n is a 1-5 digit non-zero integer line count. VETO is re-initiated after n lines have been processed.
Nn	do not perform the operation on the next n lines (including the current line), where n is a 1-5 non-zero digit integer line count. VETO is re-initiated after the n lines have been processed.
Ln	perform the operation without VETO on the next n lines (including the current line), where n is a 1-5 digit non-zero integer line count. All lines following the current line that are affected by the operation will be listed in full. VETO is re-initiated after n lines have been processed.

In addition to the above VETO options, you may enter a substitute line for the line in question. Do this by typing an equal sign (=) followed by the new text line. The line is entered according to the current SYSTEM and tab characters may be used. After you have entered the line, EDITOR will echo it and query again for one of the above legal VETO responses. When you give an affirmative response, the new text line is added to the EWFILE. This type of VETO response may not be used during the processing of a LIST, LISTF or DELETE directive.

3.4.6 FROM n (AT n or TO n)

The 'FROM n' parameter defines a starting line number (n must be a legal line number, as defined in Section 3.2.2). It can be used with the following directives: DUP, INSERT, MERGE, MOVE, OLD, RESEQ. FROM n may be abbreviated 'FR n.'

Examples:

```
OLD A FROM 10 BY 1.

RESEQ FROM 100 BY 10.

MOVE 100-500 TO 1000.

DUP 20 AT 60, 110.
```

3.4.7 BY m

The BY m parameter specifies a line number increment; m defines the interval between line numbers generated while processing the following directives: DUP, INSERT, MERGE, MOVE, OLD and RESEQ. Legal values for m are discussed under each directive.

3.5 Editing Systems

You can simplify the job of entering text lines at the terminal by taking advantage of the formatting systems provided by EDITOR. The SYSTEM directive sets work file attributes to conform to certain standards. This affects lines entered at the terminal or read from a file using the READ directive. For example, SYSTEM FORTRAN examines a line to determine whether it is a comment, a continuation line, a numbered text line or an unnumbered text line. Once this is determined, the line is reformatted to conform to FORTRAN language conventions. This saves you the effort of indenting text on a line by line basis. If you do not specify an editing system, 'SYSTEM,FORTRAN' is assumed.

The format of the SYSTEM directive is as follows. Note that the parameters are order dependent.

SYSTEM,sysname,[cmdlfn],[UPDATE]

cmdlfn — the name of an exec (control statement) file. The file names UPDATE or UP are not valid names for cmdlfn. This parameter is used to reference an exec file for use with the GO directive. When the GO directive is processed, the commands contained in cmdlfn will be executed after EWFILE is saved. Using SETFILE as cmdlfn allows the exec file directive to be in EWFILE because GO initiates 'SAVE,SETFILE.'

UPDATE — causes all the compiler directives to call UPDATE before the compiler is called (see Section 3.10).

sysname — specifies one of the formatting systems: GENERAL, TEXT, BASIC, FORTRAN, COMPASS or BATCH as described below.

3.5.1 BASIC

The BASIC editing system uses the BASIC language conventions to set up EDITOR work file attributes. EDITOR line numbers are used as BASIC statement numbers. You need not supply an additional number.

The line number is included as part of the text line and begins in column 1. For this reason, only integer line numbers can be used under SYSTEM,BASIC. No other reformatting occurs. Line numbers may still be terminated by an equal sign (which will not appear in the text line). SYSTEM,BASIC sets the maximum line length to 140 columns. For example, typing:

```
10INPUT X
20 Y=X*2
30=PRINT"TWO TIMES"X;"IS"Y
```

produces the following text lines in EWFILE.

```
line 10:    10INPUT X
line 20:    20 Y=X*2
line 30:    30PRINT"TWO TIMES"X;"IS"Y
```

3.5.2 BATCH

The BATCH editing system includes some shortcuts to help set up jobs for batch processing.

No reformatting occurs. This is identical to SYSTEM GENERAL except for the following:

1. LENGTH is set to 80.

2. The characters "*JOBCARD*" when found in columns 1-9 during a SAVE operation, will be automatically changed to "userid, PNproblem-number". Additional parameters may be added to the job card (e.g. "*JOBCARD*,JC500,RG1.").

3. Both the BATCH and GO directives will cause the EWFILE to be saved and disposed to the input queue.

There is an example of an SPSS job disposed for batch processing in Section 3.11.

3.5.3 COMPASS

The COMPASS editing system uses the COMPASS language conventions to set up the EDITOR work file attributes.

No formatting is performed. However, the following actions are taken when SYSTEM COMPASS is requested:

1. LENGTH is set to 72.

2. TAB stops are set at columns 11, 20 and 36 (unless tab stops are already set).

During intra-line editing, overflow lines are broken at column 72 and continued on a subsequent line with a comma in column one and the overflow beginning at column two.

3.5.4 FORTRAN

The FORTRAN editing system sets up the EDITOR work file attributes and reformats lines according to FORTRAN language conventions.

The text line begins immediately following the line number and is structured according to the following rules:

1. Comment lines.

   a. If the first text character is a "C" and it is not followed immediately by any alphanumeric character or an equal sign or a left parenthesis, the line is treated as a comment line and no reformatting will be performed. This enables normal FORTRAN statements beginning with a C to be entered normally.

For example, the lines:

```
230=C      THIS IS A COMMENT
240=C*   AND SO IS THIS
250=C+  THIS TOO
260=C (I)=J*K
270=C = J
```

are all entered into the EWFILE as FORTRAN comments with no reformatting performed. The following:

```
120=COMMON A,B,C
130=CALL NOBLANK
140=C(I) = J*K
150=C= J
```

would be treated as normal FORTRAN statements and reformatted according to the standard FORTRAN conventions.

b. If the first character is an asterisk (*), the line is always treated as a comment line and no reformatting is performed.

2. Continuation lines.

If the first non-blank character in columns 1 to 6 is a "+" (plus sign), the line is treated as a FORTRAN continuation line and is shifted so that the plus sign appears in column 6. During intra-line editing, overflow lines are broken after column 72. A continuation line is made with "+" in column 6 and the next character in column 7. For example, typing:

```
650 100FORMAT (1H1, 25HTHIS FORMAT STATEMENT IS
660+35HLONG ENOUGH TO REQUIRE CONTINUATION)
```

produces the following text lines:

```
      Col 1     Col 7
      ↓         ↓
line 650: 100   FORMAT(1H1, 25HTHIS FORMAT STATEMENT IS
line 660:      +35HLONG ENOUGH TO REQUIRE CONTINUATION)
```

3. Unnumbered text lines.

If the first non-blank text character occurring before column 7 is not a number, the text line is shifted so that the character appears in column 7. For example, typing:

```
100PROGRAM FORTRAN (INPUT,OUTPUT,TAPE1=INPUT)
110=RATS=0.0
120CALL OOPS(RATS)
```

produces the following text lines in EWFILE:

```
          Col 1       Col 7
            ↓           ↓
line 100:             PROGRAM FORTRAN (INPUT,OUTPUT,TAPE1=INPUT)
line 110:             RATS=0.0
line 120:             CALL OOPS(RATS)
```

4. Numbered text lines.

   If the first non-blank text character occurring after the line number, but before column 7, is a number, the line is interpreted as beginning with a FORTRAN statement number. The first string of up to five digits is reformatted to begin in column 1 and the next non-blank, non-numeric character occurring before column 7 is placed in column 7. Consider the following entries:

```
1002000FORMAT (A10)
100=2000FORMAT (A10)
100 2000 FORMAT (A10)
110 2000      FORMAT (A10)
```

   The first line is interpreted as an unintentional mistake as there is no delimiter between the EDITOR line number and the FORTRAN statement number. EDITOR will issue an error diagnostic:

   TEXT LINE SKIPPED, NUMBER OUT OF RANGE: 1002000FORMAT (A10)

```
          Col 1    Col 7
            ↓        ↓
line 100: 2000     FORMAT (A10)
```

   However, in line 110, the word FORMAT does not occur until the tenth column, which produces the text line:

```
line 110: 2000       FORMAT (A10)
```

   SYSTEM FORTRAN sets the maximum line length to 72 columns. MARGIN has no effect in SYSTEM FORTRAN.

3.5.5 GENERAL

The GENERAL editing system is used to enter control statements, text and data. No reformatting occurs. Column 1 of the text is taken to be the character which immediately follows the line number. SYSTEM GENERAL does not alter the current line length (i.e. any portion of the line which extends beyond the maximum length is truncated) by the SAVE, LIST and LIST directives). Overflow lines are not broken during intra-line editing — they are left as overflow lines and a warning message is issued.

3.5.6
TEXT

The TEXT editing system is used for text and data entry. No reformatting occurs. Column 1 of the text is taken to be the character which immediately follows the line number. This is identical to SYSTEM GENERAL except for the treatment of continuation lines created by intra-line editing. The line may extend past the current maximum line length, but not beyond column 140 without being truncated. Overflow lines are broken by dividing the line at the last blank character before the LENGTH limit. If there is no blank, the line is broken at LENGTH. This functions as a crude word processor and can be useful for entering large blocks of program comments.

3.6
Text Line Formatting Directives

There are several directives which affect the format of a text line. The SYSTEM directive sets work file attributes in terms of programming language conventions, or common uses of text processing. These attributes can be set individually. The relevant attributes are left margin, line length, tab stops and the tab character.

3.6.1
Setting Maximum Line Length — LENGTH

The LENGTH directive establishes the maximum line length, in columns, for each text line entered.

LENGTH,c.

c the line length; where c is an integer, ranging from 1 to 140, which denotes the last column of the line.

When EWFILE is created, c has the default value 72, or it takes the previous value of line length if you had retained EWFILE at the end of an earlier interactive session. When text lines are entered into EWFILE from a terminal or read from a coded file, up to 140 characters may exist in each line regardless of the line length setting, but EWFILE text lines are truncated to the current LENGTH specification during output operations, such as SAVE and LIST. The directives, OLD, INSERT and MERGE, truncate the line to c characters upon input. You may reset LENGTH at any time without altering the existing contents of EWFILE; however, files subsequently created from EWFILE may contain truncated lines if any text lines had been entered under a greater line length setting. A warning is always given if lines are truncated by the SAVE directive.

When processing ASCII files containing non-printing characters while %SHOWNPC (see Section 8.3.11) is set 'ON' or 'PART', the length of the printed line may be longer than the number of characters set by the LENGTH directive. The symbols used to represent non-printing characters may require additional characters. Note that the number of text characters represented will match the LENGTH setting.

3.6.2
Setting the Left Margin — MARGIN

The MARGIN directive sets an automatic left margin for text lines entered from the terminal under systems GENERAL, TEXT, BATCH and COMPASS, but has no effect under SYSTEM,BASIC or SYSTEM,FORTRAN.

MARGIN[,c].

c a 1-3 digit integer which specifies the column in which text lines are to begin. The value of c must be less than the current LENGTH setting.

When EWFILE is created, c has the value 1, or it takes the previous value of MARGIN if the user had retained EWFILE at an earlier logout. Just as with the reformatting performed under SYSTEM,FORTRAN, no carriage movement occurs, but each text line is internally shifted so that the first text character appears in the column denoted by c. No margin reformatting is done by OLD, INSERT or MERGE.

Typing: 'MARGIN', 'MARGIN,0.', or 'MARGIN,1.' are equivalent and set the left margin to column 1.

3.6.3
Defining Tab Stops — TAB

The TAB directive sets up to seven tab stops. EDITOR clears all previously defined tab stops each time TAB is entered.

TAB[,c_1][,c_2]...[,c_7].

$c_1,c_2,\ldots,c_7$ 1-3 digit integers which denote the columns in which tab stops are set. $c_1,c_2,\ldots,c_7$ must appear in ascending order. If no stops are specified (i.e., TAB.) all tab stops are cleared.

3.6.4
Defining Tab Characters — TABCH

The TABCH directive defines a tab character. When a tab character is encountered in any text line entered from the terminal, the text is internally spaced over to the next tab stop. If the tab character occurs beyond the largest currently defined tab stop, it is processed as a text character. When a new EDITOR work file is created, the default tab character is the semicolon (;).

TABCH[,char].

char any character (including upper/lower case characters, special or control characters) specifying the tab character. EDITOR takes char to be the first character following the delimiting comma or blank. If "TABCH." is typed, no tab character is defined. Note that the tab character is an ASCII character (e.g., 'TABCH,a.' is not the same as 'TABCH,A.').

Example:

```
SYSTEM,TEXT. MARGIN,5.
TABCH,@. TAB,15,30.
```

These directives establish work file attributes suitable for entering general text. Text cannot begin before column 5. The tab character is an at-sign, and the tab stops are in columns 15 and 30. Below is sample text:

```
100=Memo:
110=@
120=The characters are divided into 45 fonts.
130=These are listed below with the range of
140=character numbers for that font.
150=@
160=RANGE@FONT@STYLE
170=@
180=1-26@cartographic@Roman
190=501-526@simplex@Roman
```

which produces in EWFILE:

```
                Col. 5          Col. 15             Col. 30
                  ↓               ↓                   ↓
line 100:       Memo:
line 110:
line 120:       The characters are divided into 45 fonts.
line 130:       These are listed below with the range of
line 140:       character numbers for that font.
line 150:
line 160:       RANGE     FONT              STYLE
line 170:
line 180:       1-26      cartographic      Roman
line 190:       501-526   simplex           Roman
```

3.7 Getting Lines into EWFILE

EWFILE is initially empty. Text lines may be entered just by typing lines that begin with line numbers. Thus, text lines are a special class of EDITOR directives, in that any type-in which begins with a digit instructs the system to call EDITOR, which will check for a valid line number and format the remainder of the line into EWFILE. Text lines are not sent to EDITOR immediately. First, they are stored in an input buffer. Text is not processed by EDITOR until an EDITOR directive or a system command is issued. Entering an EDITOR text line does not cause text to be saved.

All text lines typed at the terminal are entered in ASCII. This does not adversely affect you when operating in DC mode because the lines are translated to Display code before use (e.g. by a SAVE or LISTF directive). While working in DC mode any text you list at the terminal is automatically folded to upper-case (a 63 character ASCII subset) for display. If you are working in DC mode you will see and use upper-case characters. If you are working in AF mode you will see and use the full ASCII character set.

When you are entering text lines on an individual basis, a carriage return and linefeed are sent to the terminal to prompt you for input. When you are using auto-line numbering, a line number is sent to the terminal to prompt for input.

Alternatively, you may build a work file by:

1. renaming a previously created work file,
2. reading in a Display code file, or
3. reading in a paper tape or magnetic tape cassette.

Once a work file has been constructed, any of three output operations may be performed:

1. copy the contents of EWFILE into a Display code file (using the directives: SAVE or LISTF);
2. list the contents of EWFILE at the terminal (using the directive, LIST);
3. punch text line images on paper tape (using the directive, PUNCH).

3.7.1 Automatic Line Numbering — N

This command initiates automatic line numbering. N is an interactive command and not an EDITOR directive.

N[,n][,m].

n an optional starting line number. Permissible values range from 0 to 99999.99. If omitted, line numbering begins following the last automatic line number produced; or, if the N directive was not used previously during the session, line numbering begins with 100. If the auto-line number value exceeds 100000.0, a warning message is given and auto-line numbering stops.

m an optional line number increment. Permissible values range from 0.01 to 100.00. If omitted, successive line numbers are incremented by the last specified value of m; or, if not previously specified during the session, successive line numbers are incremented by 10.

When entered, the system outputs the starting line number followed by an equal sign. You may then enter the text for that line. After each line is entered, the computer responds with the next line number followed by an equal sign. Automatic line numbering may be terminated by typing an equal sign followed by a carriage return. Although typing ESCAPE will also terminate automatic line numbering, use of the escape key is discouraged because it may abort EDITOR and cause loss of typed text.

All text lines entered are saved and transmitted to EDITOR when you terminate automatic line numbering. (See below.)

```
N.
100=THIS IS MY TEXT
110=IT SURE IS EASY
120=
```

READY 08.23.55

After the system outputs each line number at the terminal, it increments the value of n by the value of m, i.e., the value of n is always the next line number to be issued. Thus, if line numbering is interrupted, it may be resumed where it ended by typing: N. However, if you have entered line numbers manually during the interruption, you must re-specify the value of n if automatic line numbers are to begin following the last line entered manually.

Examples:

```
N.
100=PROGRAM READ (OUTPUT, TAPE)
110=READ (1,100) IN
120=←CR

READY 08.26.31

300=100 FORMAT (A10)
```

Now, if you want automatic line numbering to continue following line 300, type:

```
N,310.
```

But to continue entering lines after the last line number type:

```
N.
```

You may override a line number by prefixing a type-in with an equal sign. If the next character is a number, then the type-in is interpreted as a text line and automatic numbering continues. Otherwise, the type-in is processed as a directive or as a command and automatic line numbering terminates.

In the example below, line 40 is overridden in order to insert line 15. Line number 50 is also overridden and automatic line numbering terminated by issuing a SCOPE/HUSTLER control statement.

In the example below, the user overrides line 40 in order to insert line 15. She also overrides line number 50, and terminates automatic line numbering by issuing a SCOPE/HUSTLER control statement.

```
N,10,10.
10=PROGRAM EXAMPLE (INPUT,OUTPUT,TAPE1=INPUT)
20=READ(1,100) IN
30=100 FORMAT (1X,4A10)
40=←15 DIMENSION IN (4)
40=PRINT 100, IN
50=←ATTACH, TAPE1,DATAFILE,PW=SECRET.
    ATTACH,TAPE1,DATAFILE,PW=*---*.

READY 16.42.12

N.

50=END
60=←CR
EON-PROCESSING TEXT
READY 16.42.20
```

The line number sequence generated by "N." is not associated with any particular work file. For example, assume text is being entered into one EWFILE with the aid of auto-line numbering. After editing, a new work file is brought in and you type "N." to start numbering lines for this new file. The first line number supplied by auto-line numbering will be the next in the sequence generated for the preceding EWFILE. It is your responsiblity to keep track of line numbering.

If the N command was initially typed interspersed with other commands or directives the way in which you leave the auto-numbering facility will affect whether or not commands following N will be executed.

There are five ways of leaving the auto-numbering facility; these are:

1. typing the abort character, (default = ESC),
2. typing an equal sign (=) followed by a carriage return,
3. typing an equal sign followed by a series of commands or directives,
4. hanging up the phone, or
5. typing %QUIT.

Correspondingly, the following will happen:

1. Any remaining commands or directives are processed. This is not recommended. It may cause text to be lost.
2. Any remaining commands or directives are processed.
3. The series of commands or directives following the equal sign are processed and those given earlier are ignored.
4. You are logged out without executing any additional commands. All that had been typed in may be lost.
5. You are logged out without executing any additional commands. All that had been typed in may be lost.

For example, a user types:

N.FTNX.

The system will prompt for text lines to be entered. If, when you have finished typing in the program, you use the abort character, then the FTNX directive will compile and execute the program. If you instead typed '=ASSETS' to exit from auto-numbering, the ASSETS command would supersede the earlier FTNX command given.

When you leave auto-numbering mode by one of the first two means listed above, the message:

EON-PROCESSING TEXT

appears on your terminal. (EON stands for 'end of numbering.') If you had not entered any text lines while in auto-numbering mode, the following informative message is given:

NO TEXT LINES HAVE BEEN ENTERED

3.7.2
Reading Numbered Text Lines and EDITOR Directives — READ

The READ directive reads the specified coded file, entering numbered text lines into EWFILE. EDITOR directives are processed as they are encountered. The file must contain line images in exactly the same format in which they would appear if typed at the terminal. Because the input is handled just as though it were entered from the terminal, EWFILE need not be empty before executing this directive. Lines may use either the Display code or ASCII character set. Note that the N directive is not legal in a READ file.

READ,lfn[,NR].

lfn the name of a standard SCOPE coded file which contains EDITOR directives and text lines in the format described above.

NR lfn is not rewound before the read operation.

The READ file may contain any mixture of AF lines and Display code lines, and the AF parameter is not necessary to specify it — EDITOR figures out the character set of each line or directive it reads.

EDITOR directives and text are read from lfn until end-of-section or until another READ directive is encountered. Other directives may follow on the same line as the original READ directive. When all directives have been processed, the READY or OK message will be displayed at the terminal.

3.7.3
Reading Text Lines from a File — OLD

This directive enters text lines into an empty EWFILE from the file designated by lfn. Text lines can be entered from any standard SCOPE coded file by instructing EDITOR to assign them sequential line numbers generated according to n and m. On the other hand, when lfn consists of lines which already contain line numbers in a suitable format, you may retain those same numbers as EDITOR line numbers in EWFILE.

OLD,lfn[,FROM n][,BY m][,AF][,CASE][,CTRL][,NR][,VETO][,txt][,UPDATE]

lfn the name of a local file in standard SCOPE coded format. This file, which is not altered by OLD, is read until end-of-information. All end-of-section and end-of-partition marks encountered, except those occurring at the end, are entered in EWFILE as the special text lines *EORnn and *EOF. Trailing end-of-section and end-of-partition marks are dropped.

FROM n the starting value for generated line numbers. If both 'FROM n' and 'BY m' are omitted, line numbers are taken from lfn. If just FROM n is omitted, generated line numbers begin with 100.

BY m a line number increment for generated line numbers (as described in Section 3.4.7). If 'BY m' is omitted when 'FROM n' is specified, successive line numbers are incremented by 10.

AF allows you to read an ASCII file. The default is to read the file in Display code, but OLD automatically performs 'SET,AF=ON' if the file is ASCII. A warning message is issued if the file appears to be the wrong character set. The warning turns VETO on so you may correct the problem.

CASE used in conjunction with the txt and AF parameters, specifies that the case of the search string must match (e.g. /Abc/ is not the same as /abc/). The default does not consider case (e.g. /Abc/ is the same as /abc/). CASE may be abbreviated C.

CTRL if AF is specified, CTRL allows you to search for strings containing control codes. The CTRL parameter is meaningless unless txt and AF are also specified.

NR lfn is not rewound before the operation.

VETO specifies each text line is to be displayed at the terminal before it is entered. You then enter one of several execution options (as described in Section 3.4.5). VETO may be abbreviated V.

txt a character (search) string may be supplied to select only certain lines to be entered into the work file. Only those lines which match the character string will be entered into the EWFILE. see Section 3.4.3 for a full description of the txt parameter.

UPDATE parameter indicates that the file lfn is an UPDATE COMPILE-file. The COMPILE-file must be generated by UPDATE **without** the "D" or "8" option on the UPDATE control statement to insure that full UPDATE identifiers and sequence numbers are written on the file beginning at column 74.

When UPDATE is used, 'FROM n' must be specified.

Use of the UPDATE parameter allows EDITOR to create an UPDATE correction set which reflects any changes subsequently made to the text in more details. UPDATE may be abbreviated UP.

When neither (c_1,c_2) nor 'FROM n' and 'BY m' are specified, lfn must contain lines numbered as follows: (a) under SYSTEM BASIC line numbers must appear in columns 1 through 5; (b) otherwise, line numbers must appear within the 14 columns just beyond the current maximum length; this is the format created by the SAVE directive. (Note that in (b) LENGTH must be set to the same value in effect when the file was created by SAVE.)

Alternatively, you may specify the column range in which line numbers appear, or you may specify starting and increment values for line numbers to be generated. Whenever line numbers are taken from the line itself, EDITOR re-orders the incoming text lines to maintain line numbers in ascending order. However, when lines are assigned generated line numbers, they enter EWFILE in the same order that they appear in file lfn.

When EDITOR is instructed to read line numbers from the text, a line number which does not conform to the rules of Section 3.2.2 causes EDITOR to produce the diagnostic:

```
*MISSING LINE NUMBER; YES=n
?
```

where n is a valid line number generated by EDITOR. You then enter one of the following characters:

Y assign n to the text line in question.

N ignore the line in question.

S stop execution of the OLD directive.

To prevent monotonous repetition of this diagnostic, it is usually best to stop execution and re-examine the specified column range for superfluous characters.

If EWFILE is not empty when OLD is entered, EDITOR responds:

```
*EWFILE WILL BE RETURNED, AND A NEW ONE CREATED -
 TYPE Y TO CONTINUE*
?
```

When you type 'Y', EDITOR performs SCRATCH before executing OLD, if 'N' is typed, EDITOR stops the operation so that you may dispose of the current work file in some other way. To add text lines to a non-empty EWFILE, see the MERGE and INSERT directives (Sections 3.13.5 and 3.13.6).

Examples:

1.
```
LENGTH,72.
OLD MYFILE.
```

In this example, the OLD directive enters text lines into EWFILE from MYFILE and assigns them the line numbers appearing in columns 73 through 86 of each line. This form might be used if MYFILE were created with the SAVE directive under the same LENGTH specification.

2.
```
OLD COMPILE (81,86)
```

Here, text lines are entered from COMPILE and assigned the line numbers appearing in columns 81 through 86.

3.
```
COPYBR,DATA,DUMMY.
OLD DATA,NR,FROM 10.
```

Text lines are entered from DATA and assigned the line numbers 10, 20, 30, etc. DATA is not rewound in order to bypass the first section.

4.
```
OLD COMPILE,FROM 10 BY 10,UPDATE.
```

This causes the file COMPILE to be entered as special UPDATE lines in EWFILE.

Character Set Warning

EDITOR checks the file (lfn) to make sure it appears to be in the correct character set. If it detects a line in the wrong character set, it issues a warning message like this:

WARNING - DISPLAY-CODE DATA EXPECTED ON FILE lfn MAY BE ASCII

Then EDITOR turns on VETO, allowing you to see the line in question, and decide whether to continue (C) or abort (S or K) the command. The warning will be issued only once per command.

A correctly formatted AF file will never be mistaken for Display code by EDITOR; however, certain rather unusual Display code text lines may look to EDITOR like AF, and trigger this warning erroneously. In this case, simply type "C" and EDITOR will continue. EDITOR will make this mistake on a Display code line that is 9 characters or longer, in which columns 1, 3, 5, 7 and 9 contain only the characters "A", "B", and "C".

3.7.4 Reading Paper Tape/Floppy Disk

There are three commands which affect tape or disk input into an EDITOR work file. Note that any tape operation described below will also work for a floppy disk.

TAPE TAPEC	SYSTEM commands
% READER	a Front-end command

Each line entered from tape must be terminated by a carriage return.

After receiving a tape command, the message 'READY FOR TAPE' is printed at the terminal; you then start the tape through the tape reader at the terminal. After the tape has been read, you must indicate end-of-tape (EOT) by entering the abort character (default=ESC). After an escape, wait for the EOT message before sending more data. This is especially important when using minicomputers and "smart" terminals that transmit volumes of data at high speeds.

TAPE and TAPEC read from tapes into EWFILE, instructing EDITOR to enter the lines in the same format as lines entered manually. EDITOR accepts only text lines under TAPE, but processes both text lines and EDITOR directives under TAPEC.

Text lines entered with the TAPE directive must begin with a valid line number. Because the handling of text lines entered from tape is no different from that of text lines entered manually, EWFILE need not be empty.

With TAPEC, all lines that begin with a number are treated as text lines; all others are processed as EDITOR directives. The actual entering of text lines into EWFILE and the processing of directives occur after you indicate end-of-tape. If there is an error in one of the directives, none of the remaining lines is processed. EWFILE need not be empty before the TAPEC command is given.

To use these commands in an exec file the following sequence must be used.

```
MISTIC.
{TAPE|TAPEC}.
END.
EDITOR.
END.
```

Without the 'EDITOR.', 'END.' sequence the file will not be written after the tape is read.

The %READER command allows the Front-End computer to automatically start and stop input. Paper tapes may be read reliably only from terminals equipped with a tape reader which can be controlled in this fashion. When the Front-End computer can no longer accept additional input, input is halted by transmitting a DC3 character to the terminal. Input is resumed by transmitting a DC1 character to the terminal. This ensures that information will not be lost. See Section 8.3.13.

3.8 Outputting the Contents of EWFILE to a File

The contents of EWFILE can be output either in EWFILE format or in standard coded format.

3.8.1 Listing the Contents of EWFILE at the Terminal — LIST

The LIST directive lists the contents of the specified lines of EWFILE at the terminal; if no parameters are specified, all text lines are listed. Each line listed is preceded by its EDITOR line number followed by "=" (equal sign); e.g.,

100=THIS IS THE TEXT (except in SYSTEM,BASIC).

FULL is the default value, meaning that lines are listed without suppression of blanks. Optional parameters enable you to list only selected lines, to omit EDITOR line numbers, or to suppress extra blanks.

Also, unless FULL is specified, lines appear in compressed format, such that two or more consecutive blanks are reduced to a single blank. Optional parameters enable you to list only selected lines, to omit EDITOR line numbers, and to list text lines in their actual format.

LIST[,lnum|@[abb]][,txt][,AF][,CASE][,CTRL][,NOSEQ][,FULL][,NFULL][,UNIT][,VETO]

lnum — the line numbers and line number ranges (as defined in Section 3.4.1) to be listed.

@abb — causes the character string for which abb is an abbreviation to be listed. If @ with no abb is specified, then all strings with currently defined abbreviations are listed. See Section 3.16 for a detailed discussion of string abbreviations.

txt — a text search string (as defined in Section 3.4.3). Only lines containing the specified character string are listed. If lnum is specified, the text search is restricted to the specified lines.

AF — causes the file to be written in the full ASCII character set, without folding to the 63-character subset.

CASE — used in conjunction with the txt and AF parameters, specifies that the case of the search string must be matched (e.g. /Abc/ is not the same as /abc/). The default does not consider case (e.g. /Abc/ is the same as /abc/). CASE may be abbreviated C.

CTRL — if AF is also specified, CTRL causes all control codes to be written as they appear in the work file. CTRL also allows you to search for strings containing control codes.

NOSEQ — specifies that EDITOR line numbers are to be omitted from the listing. NOSEQ may be abbreviated NS.

FULL — specifies that text lines are to appear without suppression of multiple blanks, listing lines in their actual format. FULL is the default value and may be abbreviated F.

NFULL — specifies that lines appear in compressed format, such that two or more consecutive blanks are reduced to a single blank. NFULL may be abbreviated NF.

UNIT specifies that the search string, txt, must occur as a unit (as described in 3.4.4). UNIT may be abbreviated U.

VETO specifies that execution is to pause after each line is listed. You then enter one of the options described in Section 3.4.5. If Nnn is typed, EDITOR skips the next nn lines before continuing the listing. If Ynn is typed, listing proceeds for nn lines before pausing again. VETO may be abbreviated V.

Examples:

Suppose the following lines are entered into an empty EWFILE:

```
100PROGRAM HITHERE (OUTPUT)
110C THIS PROGRAM PRINTS 'HI THERE'
120PRINT 100
130 100FORMAT(9H HI THERE)
140END
```

The examples below show various uses of the LIST directive followed by their output.

1.
```
OK=LIST,NF.
100= PROGRAM HITHERE (OUTPUT)
110=C THIS PROGRAM PRINTS ``HI THERE''
120= PRINT 100
130=100 FORMAT(9H HI THERE)
140= END
```

2.
```
OK-LIST,100:+1,FULL,NOSEQ.
PROGRAM HITHERE (OUTPUT)
C THIS PROGRAM PRINTS 'HI THERE'
```

3.
```
OK=LIST,/100/.
120=      PRINT 100
130=100   FORMAT(9H HI THERE)
```

4. Consider the following BASIC program entered in AF mode:

```
OK-set,af=on;
OK-system,basic. n;
100=rem A program to find the average age of a group of students.
110=rem Initialize counters: c=no. people; a=total age.
120=let c=0
130=let a=0
140=rem Input an age
150=input n
160=rem Test for end of data:  n=0
170=if n=0 then 210
180=print n
190=rem Go to next age input
200=go to 150
210=rem Calculate average age
220=let b=a/c
230=print 'Average age of',c,'students=',b
240=end
250== CR
```

```
OK-listref
100rem A program to find the average age of a group of students.
110rem Initialize counters:  c=no. people, a=total age.
120let c=0
130let a=0
140rem Input an age
150input n
160rem Test for end of data:  n=0
170if n=0 then 210
180print n
190rem Go to next age input
200go to 150
210rem Calculate average age
220let b=a/c
230print "Average age of",c,"students=",b
240end
OK-listcbrief
100REM A PROGRAM TO FIND THE AVERAGE AGE OF A GROUP OF STUDENTS.
110REM INITIALIZE COUNTERS:  C=NO. PEOPLE, A=TOTAL AGE
120LET C=0
130LET A=0
140REM INPUT AN AGE
150INPUT N
160REM TEST FOR END OF DATA: N=0
170IF N=0 THEN 210
180PRINT N
190REM GO TO NEXT AGE INPUT
200GO TO 150
210REM CALCULATE AVERAGE AGE
220LET B=A/C
230PRINT "AVERAGE AGE OF",C,"STUDENTS=",B
240END
```

Notice that NAF causes the file to be listed in upper case only.

3.8.2 Copying EWFILE to a Standard File — SAVE

The SAVE directive writes the contents of EWFILE onto the file indicated by lfn in standard SCOPE coded format (see Chapter 4 of the*SCOPE/HUSTLER Reference Manual*). EWFILE is not changed. Optional parameters lnum and txt enable you to write selected text lines onto the file lfn. Unless NOSEQ is specified, EDITOR line numbers are appended to each line, beginning in the column just beyond the current LENGTH limit. Thus, the contents of file lfn may be re-entered into EWFILE by simply typing: OLD,lfn (provided the same LENGTH setting is in effect).

EDITOR converts the special text lines, *EOSnn (*EORnn) and *EOP (*EOF) into SCOPE end-of-section and end-of-partition marks, respectively, in file lfn.

SAVE,lfn[,lnum|@[abb]][,txt][,AF][,CASE][,CTRL][,NOSEQ][,NR][,UNIT]
[,SOURCE][,UPDATE]

lfn — the name of a file onto which text lines are written in standard SCOPE coded format.

lnum — the line numbers and/or line number ranges (as defined in Section 3.4.1) of the text lines to be written to file lfn.

@abb — when this parameter is used, the SOURCE parameter must also be listed, it causes the specified string abbreviations to be saved on file lfn. If @ appears without abb, then all strings with abbreviations will be saved. See Section 3.16 for the details of abbreviation usage.

txt — a text search string (as defined in Section 3.4.3). Only text lines containing the specified character string are written to lfn. If lnum is also specified, the text search is restricted to the lines indicated.

AF — causes the lines specified by lnum to be written in the ASCII character set, without folding lower-case characters to upper case. SAVE automatically performs 'SETCODE,outlfn=AF' if all data written by this SAVE directive is ASCII. (If 'NR' has been specified, there may already be non-ASCII data on earlier sections of the file.)

CASE — used in conjunction with the txt and AF parameters, specifies that the case of the search string must match (e.g. /Abc/ is not the same as /abc/). The default does not consider case (e.g. /Abc/ is the same as /abc/). CASE may be abbreviated C.

CTRL — if AF is also specified, CTRL causes all control codes to be written as they appear in the work file. Also, control codes will not be ignored during search string processing.

NOSEQ — specifies that line numbers are to be omitted from the lines written to file lfn. NOSEQ may be abbreviated NS.

NR — lfn is not rewound. Normally the file is rewound both before and after the SAVE operation.

UNIT — specifies that the text search string must occur as a unit (as defined in Section 3.4.4). UNIT may be abbreviated U.

SOURCE — all lines output to file lfn will have the line number (or string abbreviation) preceding the text. This overrides the SEQ|NOSEQ parameter. The file lfn is in a form to be read back into EWFILE using a READ directive. String abbreviations are saved as STRING commands. SOURCE may be abbreviated SO.

UPDATE — this parameter causes an UPDATE correction set to be constructed and output to the file lfn. It is meaningful only if the EWFILE were created using an OLD directive with the UPDATE option. No *IDENT directive is automatically generated. If one is desired, it should be entered as a text line preceding the first UPDATE input line. UPDATE may be abbreviated UP. See Section 3.20 for a more complete description of UPDATE usage.

Examples:

1.
```
LENGTH,72.
SAVE,MYFILE.
```

All text lines are written to MYFILE, with their EDITOR line numbers beginning in column 73. Any text appearing beyond column 72 is not written to MYFILE.

2. SAVE,MYFILE,*F-49,NOSEQ.

   Suppose EWFILE contains text lines numbered from 1 to 200 in increments of 1. This entry writes the first 49 lines to MYFILE without their EDITOR line numbers.

3. Suppose EWFILE contains a FORTRAN program; then typing either:

   SAVE,MYFILE,/PRINT/,NR,UNIT. or,
   SAVE,MYFILE,/PRINT/U,NR.

   writes only PRINT statements to MYFILE. MYFILE is not rewound either before or after the operation. EDITOR line numbers are appended to each line.

4. SAVE,MYFILE,/FORMAT/(7,8),100-150,NR.

   Text lines from line 100 to line 150 are searched for the string, FORMAT, which must begin in either column 7 or 8. If a match is found, the entire text line is written to MYFILE along with its EDITOR line number. If used in conjunction with Example 3, the FORMAT statements are placed after the PRINT statements on MYFILE, and separated from each other by an end-of-section.

5. SAVE,ADDRESS,AF.

   In this example, EWFILE contains a mailing list which is written on ADDRESS in upper/lower case. (Normally, if you were processing ASCII data you would 'SET,AF=ON' at the beginning of your interactive session. However, if you neglected to do so, you can specify AF mode at this stage without loss of case.)

3.8.3 Listing Text Lines onto a File — LISTF

The LISTF directive lists text lines onto a file in EDITOR work file format.

This is equivalent to LIST with the following exceptions:

1. the listing is produced on file lfn instead of the file TTYTTY.

2. each line output to file lfn is prefixed with a blank, line number and then an equal sign unless NOSEQ is specified.

The format of the LISTF directive is:

LISTF,lfn[,lnum|@[abb]][,txt][,AF][,CASE][,CTRL][,UNIT][,NOSEQ][,N R][,NFULL].

The parameters operate as for the SAVE directive (Section 3.8.2).

3.9
Cataloging/Scratching and Use of Alternate EWFILEs

The contents of EWFILE can be stored on a permanent file, and reused at a later date. When the contents of EWFILE become obsolete, the contents can be erased and replaced. Other files can be transformed into EDITOR work files, so that EDITOR can modify their contents.

3.9.1
Cataloging EWFILE

Cataloging a file makes it a permanent file which is stored on disk for a user-defined period of time, called a retention period. All attributes of the EDITOR work file are retained when it is cataloged. The CATALOG statement contains several parameters to set passwords, retention period, multi-read access, cycle and id. These parameters are discussed in full in Chapter 5 of the *SCOPE/HUSTLER Reference Manual*. We will consider a subset of those parameters here. You can catalog EWFILE using the SCOPE/HUSTLER control statement CATALOG.

Example:

```
OK-CATALOG,EWFILE,PERMEWFILE,RP=30,TK=PASSKEY.
```

NOTE: This example sets the retention period (RP) to 30 days. If RP had not been specified, the default would be 15 days. All passwords are optional. In this example, a turnkey password is set. This allows you to restrict access to your file to users who know the password. This password (PASSKEY) must be specified every time the permanent file (PERMEWFILE) is attached for use.

3.9.2
Scratching an EWFILE — SCRATCH

The SCRATCH directive returns the contents of EWFILE and creates another, empty EWFILE. If the file is not permanent it is destroyed. The attributes — SYSTEM, TAB, TABCH, LENGTH, MARGIN, GOFILE, EWFLOCK, all parameters determined by the SET directive, and the option to call UPDATE before the compiler on a compilation directive — all remain unchanged. However, if EWFILE is a permanent file, it is returned but not purged. It remains a permanent file and contains all changes made prior to the SCRATCH operation. The directive format is:

```
SCRATCH.
```

3.9.3
Using a Previously Created EDITOR Work File — USE

Once EWFILE has been cataloged as a permanent file, all subsequent modifications become permanent automatically. This eliminates the need to purge and recatalog EWFILE and reduces the chance of information being lost in a system failure. For example,

Method 1: `ATTACH,EWFILE,OLDWORK,PW=JANUARY.`

If EWFILE already exists as a local file, the system will respond with the message "LOCAL FILE NAME ALREADY IN USE." If you have not created EWFILE, by issuing any EDITOR directive, the statement will work and OLDWORK will be attached as EWFILE.

Method 2: ATTACH,A,OLDWORK,PW=JANUARY.
USE,A.

If EWFILE already exists as a local file, you should attach a previously cataloged work file indirectly. The old work file OLDWORK is attached as local file A. The USE directive renames A to EWFILE and performs the necessary initialization.

In order to edit a permanent file, the file must be attached with all (including MODIFY and EXTEND) permissions granted. EDITOR will refuse to perform directives for which the required permissions are not granted. In this case, a warning message is issued. If EWFILE has read-only permission, the only available EDITOR directives are those which do not alter the file — such as LIST, EDSTAT, SAVE, and the compilation directives.

The USE directive instructs EDITOR to use the former work file and rename it EWFILE.

USE,lfn_1[,lfn_2].

lfn_1 the name of a local file already in the work file format.

lfn_2 the file name you wish to give the current EDITOR work file.

If lfn_2 is omitted and the work file is not empty, then EDITOR will rename EWFILE to the last file name used on a USE directive to reference that work file.

Thus, to exchange EWFILE with a previous work file named OLDWORK, type:

USE,OLDWORK.

Or, to give the current EDITOR work file a different name, you might type:

USE,OLDWORK,WORK.

If lfn_2 is not specified and EWFILE was not originally accessed with a USE directive, EDITOR issues the diagnostic:

SCRATCH OR NEWNAME CURRENT EWFILE

3.10 Compilation Directives

Compilation directives give you a shortcut in preparing a job for compilation and execution. There are EDITOR compilation directives for the following languages: BASIC, COBOL, COMPASS and FORTRAN. Each language has several forms of the compilation directive, which allows you to selectively compile, debug and execute EWFILE.

Compilation directives execute a pre-defined sequence of EDITOR directives and SCOPE/HUSTLER commands, which compile and execute the contents of EWFILE. In addition, the directive, GO, gives you the option of executing your own sequence of system commands.

Compilation directives have the form:

compverb[,lnum][,txt][,UNIT][,NOSEQ]

If no parameters are specified, the entire contents of EWFILE are compiled.

compverb — any of the compilation directive verbs listed below.

language	directive verbs
BASIC	BASIC,BASICX
COBOL	COBOL, COBOLX, COBOLER
COMPASS	COMP, COMPX, COMPER
FORTRAN	FTN, FTNX, FTNER

lnum — the line numbers and line number ranges (as defined in Section 3.4.1) of the text lines to be compiled.

txt — a text search string (as defined in Section 3.4.3). Only those text lines containing the specified character string are compiled. If line is also specified, the text search is restricted to the specified lines.

The EDITOR directives BASIC, COBOL, and FTN have the same names as the system commands which call the compilers. The system will recognize BASIC, COBOL, or FTN as EDITOR directives when they appear without parameters. But if they appear with parameters, they are interpreted as direct calls to the compilers. To ensure that a command is interpreted as a SCOPE/HUSTLER command, prefix it with a $. To ensure that a command is interpreted as an EDITOR directive, prefix it with a dash (-).

All compilation directives create a standard coded file named SETFILE, which contains the text lines indicated by lnum and txt; that is, they execute:

```
SAVE,SETFILE[,lnum][,txt][,UNIT][,NOSEQ].
```

Also, all compilation directives except BASIC and BASICX disconnect OUTPUT so that the program listing is not printed at the terminal. If you choose, you may list the contents of OUTPUT with LISTTY, or dispose OUTPUT to a high-speed line printer. BASIC and BASICX also differ in that they allow you to compile AF files; as a result the AF mode parameters (AF, CASE) are legal on these directives. Use of the AF mode parameters on other compilation directives will cause the compiler or assembler to abort because of bad input data.

If the UPDATE parameter was specified on the last SYSTEM directive, then an UPDATE command is constructed and executed immediately before the compiler command. The UPDATE command is:

```
UPDATE,Q,D,L=1234,I=SETFILE.
```

The compiler command then is modified to use the file COMPILE as input. For example, under the UPDATE option the FTNER directive would perform the following:

```
SAVE,SETFILE [lnum] [txt].
REWIND,LGO.
UPDATE,Q,D,L=1234,I=SETFILE.
FTN,I=COMPILE,T.
```

ERRS.
LGO.

The descriptions in the following sections show the sequence of directives and commands invoked by each compilation directive when UPDATE is not in effect.

3.10.1 BASIC, BASICX

The two forms of the BASIC compilation directive call the BASIC compiler. Both directive verbs initiate the same process. Both BASIC and BASICX compile EWFILE.

BASIC[X][,lnum][,txt][,AF][,CASE][,CTRL][,UNIT]

The following sequence of EDITOR directives and SCOPE/HUSTLER control statements will be executed.

SAVE,SETFILE[,lnum][,txt][,CASE][,UNIT],NOSEQ.
BASIC,I=SETFILE,E=AIBCS,K=AIBCS[,AS].

The AF mode parameters act as follows:

AF causes the lines specified by lnum to be processed using the full ASCII character set. It also allows CASE and CTRL to be used with a search string. Use of AF puts ",AS" on the BASIC control statement. Data will be given to BASIC in ASCII. Also, the AF parameter causes AIBCS to be connected as ASCII Fancy.

CASE used in conjunction with the txt and AF parameters, specifies that the case of the search string must match (e.g. /Abc/ is not the same as /abc/). The default does not consider case (e.g. /Abc/ is the same as /abc/). CASE may be abbreviated C.

CTRL used in conjunction with the text and AF parameters, CTRL causes BASIC to receive control characters if AF is on, and also includes control characters in the processing of text search strings.

3.10.2 COBOL, COBOLER, COBOLX

The three forms of the COBOL compilation directive call the COBOL compiler. Each directive verb initiates different processes.

The verb COBOL compiles EWFILE and calls the debugging utility ERRS. Execution is not initiated.

COBOL[,lnum][,txt][,UNIT][,NOSEQ]

The following sequence of EDITOR directives and SCOPE/HUSTLER control statements will be executed.

SAVE,SETFILE[lnum][txt][,UNIT][,NOSEQ].
REWIND,LGO.
COBOL,I=SETFILE.
ERRS.

The verb COBOLER compiles EWFILE and calls the debugging utility ERRS. If there are no errors, the file is loaded and executed.

```
COBOLER[,lnum][,txt][,UNIT][,NOSEQ]
```

The following sequence of EDITOR directives and SCOPE/HUSTLER control statements will be executed.

```
SAVE,SETFILE[,lnum][,txt][,UNIT][,NOSEQ].
REWIND,LGO.
COBOL,I=SETFILE.
ERRS.
LGO.
```

The verb COBOLX compiles EWFILE. The file is loaded and executed.

```
COBOLX[,lnum][,txt][,UNIT][,NOSEQ]
```

The following sequence of EDITOR directives and SCOPE/HUSTLER control statements will be executed.

```
SAVE,SETFILE[lnum][txt][,UNIT][,NOSEQ].
REWIND,LGO.
COBOL,I=SETFILE.
LGO.
```

If compilation errors occur, the user may list them at the terminal by typing: ERRS.

3.10.3 COMP, COMPER, COMPX

The three forms of the COMPASS compilation directive call the COMPASS assembler. Each directive verb initiates different processes.

The verb COMP assembles EWFILE and calls the debugging utility ERRS. Execution is not initiated.

```
COMP[,lnum][,txt][,UNIT][,NOSEQ]
```

The following sequence of EDITOR directives and SCOPE/HUSTLER control statements will be executed.

```
SAVE,SETFILE[,lnum][,txt][,UNIT][,NOSEQ].
REWIND,LGO.
COMPASS,I=SETFILE.
ERRS.
```

The verb COMPER assembles EWFILE and calls the debugging utility ERRS. If there are no errors, the file is loaded and executed.

```
COMPER[,lnum][,txt][,UNIT][,NOSEQ]
```

The following sequence of EDITOR directives and SCOPE/HUSTLER control statements will be executed.

```
SAVE,SETFILE[,lnum][,txt][,UNIT][,NOSEQ].
REWIND,LGO.
COMPASS,I=SETFILE.
ERRS.
LGO.
```

The verb COMPX assembles EWFILE. If there are no errors, the file is loaded and executed.

```
COMPX[,lnum][,txt][,UNIT][,NOSEQ]
```

The following sequence of EDITOR directives and SCOPE/HUSTLER control statements will be executed.

```
SAVE,SETFILE[,lnum][,txt][,UNIT][,NOSEQ].
REWIND,LGO.
COMPASS,I=SETFILE.
LGO.
```

If compilation errors occur, you may list them by typing: ERRS.

3.10.4 FTN, FTNER, FTNX

The three forms of the FORTRAN compilation directive call the FORTRAN 4 compiler. Each directive verb initiates different processes[1].

The verb FTN compiles EWFILE and calls the debugging utility ERRS. Execution is not initiated.

```
FTN[,lnum][,txt][,UNIT][,NOSEQ]
```

The following sequence of EDITOR directives and SCOPE/HUSTLER control statements will be executed.

```
SAVE,SETFILE[,lnum][txt][,UNIT][,NOSEQ].
REWIND,LGO.
FTN,I=SETFILE,T.
ERRS.
```

If you use 'FTN,' with a comma and no parameters, SCOPE/HUSTLER will interpret this as a system command rather than an EDITOR directive. The FORTRAN 4 compiler will be invoked; however, SAVE, SETFILE, REWIND, LGO and ERRS will not be called or executed.

The verb FTNER compiles EWFILE and calls the debugging utility ERRS. If there are no errors, the file is loaded and executed.

```
FTNER[,lnum][,txt][,UNIT][,NOSEQ]
```

The following sequence of EDITOR directives and SCOPE/HUSTLER control statements will be executed.

---

[1]XFTN5 is a program which may be used to invoke the FTN 5 compiler. It effectively provides EDITOR directives for FTN 5, similar to those for FTN 4. XFTN5 resides on the Unsupported Library and is not an officially supported product of the Computer Laboratory. Further documentation of XFTN5 may be obtained using the command:

```
HELP,L*UNSUP,XFTN5.
```

```
SAVE,SETFILE[,lnum][,txt][,UNIT][,NOSEQ].
REWIND,LGO.
FTN,I=SETFILE,T.
ERRS.
LGO.
```

The verb FTNX compiles EWFILE. If there are no errors, the file is loaded and executed.

```
FTNX[,lnum][,txt][,UNIT][,NOSEQ]
```

The following sequence of EDITOR directives and SCOPE/HUSTLER control statements will be executed.

```
SAVE,SETFILE[,lnum][,txt][,UNIT][,NOSEQ].
REWIND,LGO.
FTN,I=SETFILE,T.
LGO.
```

If compilation errors occur, you may list them by typing: ERRS.

Note: FORTRAN 5 parameters may vary from FORTRAN 4. Refer to Chapter 11 in the *CDC FORTRAN Version 5 Reference Manual* for further discussion of FTN 5 control statement options.

3.11 Disposing a Job for Batch Processing — BATCH

In addition to the compilation directives, there are two other shortcuts to help you prepare jobs for execution. Both directives are intended for use with files containing SCOPE/HUSTLER commands. The BATCH directive disposes the contents of EWFILE as a job for batch processing. The GO directive executes the contents of an exec file after saving the contents of EWFILE in SETFILE. The GO directive is discussed in Section 3.12. The following description includes the format of the BATCH directive and the sequence of directives and commands which are invoked.

```
BATCH[,lnum][,txt][,UNIT][,NOSEQ]
```

The following sequence of EDITOR directives and SCOPE/HUSTLER control statements will be executed.

```
SAVE,SETFILE[,lnum][,txt][,UNIT][,NOSEQ]
DISPOSE,SETFILE,IN.
```

Example:

The following example illustrates an SPSS batch job which has been entered in EWFILE. To submit this job, you type 'BATCH.' The job is disposed to the input queue. The sequence number of the job is displayed at the terminal.

```
100=*JOBCARD*,CM55000,T30,L100,RG1.
110=ATTACH,DAT,SPSSDATA.
120=HAL,SPSS,D=DAT.
130=*EOR
140=VARIABLE LIST    AGE,INCOME,DISTANCE
150=INPUT FORMAT     FIXED(5X,F2.0,F6.0,T19,F3.1)
160=N OF CASES       UNKNOWN
170=MISSING VALUES   AGE(99,BLANK)/INCOME(BLANK)/DISTANCE(99.9)
180=PEARSON CORR     AGE TO DISTANCE
190=READ INPUT DATA
200=FINISH
OK-batch.
 SUBMITTED UNDER SEQUENCE       TB25263.
```

Note that in SYSTEM,BATCH when no exec file is specified, the directives BATCH and GO are equivalent.

3.12
Exec Files and EDITOR — GO

It is often necessary to repeatedly execute a series of commands or directives. Such repetitious tasks can be tedious and time consuming. EDITOR allows you to reduce this tedium through the automatic use of "exec files." Control statement images can be stored in exec files as modules for later use. The GO directive initiates execution of the commands stored in the exec file. Exec files are discussed in greater detail in Chapter 9.

GO[,execlfn][,lnum][,txt][,UNIT][,NOSEQ].

execlfn — the name of a coded file, which contains a sequence of system commands as described in Chapter 9. Note that using SETFILE as execlfn initiates execution of the commands at the beginning of the line range given on the GO command. This special case is useful because the commands themselves may be edited as easily as the text they work with.

If execlfn is specified, the following directive and command are executed:

```
SAVE,SETFILE[,lnum][,txt][,UNIT][,NOSEQ].
EXEC,execlfn.
```

When execlfn is omitted, EDITOR executes a different set of commands according to the following conditions:

1. If the execlfn parameter has been specified in the last SYSTEM directive entered, EDITOR executes the system commands contained in the specified file:

   ```
   SAVE,SETFILE[,lnum][,txt][,UNIT][,NOSEQ].
   EXEC,execlfn.
   ```

2. If the current editing system is BASIC, EDITOR executes the directive: BASIC.

3. If the current editing system is FORTRAN, EDITOR executes the directive: FTNER.

4. If the current editing system is BATCH, EDITOR executes the directive: BATCH.

5. If the current editing system is COMPASS, EDITOR executes the directive: COMPER.

6. If none of the above conditions exist, EDITOR issues the diagnostic:

   NO FILENAME FOR "GO" IN SYSTEM GENERAL

Examples:

1. If the local file INIT contains the initialization file for your job and EWFILE contains your job. The GO statement would be entered as follows:

   ```
   GO,INIT.
   ```

2. If the commands you wish to use as an exec file are part of EWFILE, you should use the following form in your EWFILE:

   a. Specify SETFILE as cmdlfn with the SYSTEM directive.

   b. Begin the control statement section with a SKIPF command. This causes this section to be skipped when the remainder of SETFILE is processed.

   c. Specify SETFILE as the input file where necessary.

```
OK-[redacted]
.
.
.
10=[redacted]
20=[redacted]
30=[redacted]
40=[redacted]
100=
.
.          FORTRAN program
.
180=[redacted]
1000=
.
.          data
.
```

3.13 Inter-line Editing (Line-by-Line Editing)

After text lines have been entered into EWFILE, you may wish to rearrange them, or to insert additional text lines from another file.

The RESEQ directive cannot rearrange lines, but it can facilitate other EDITOR operations by partitioning related portions of text into specified line ranges, or simply by renumbering text lines with uniformly incremented line numbers.

Relocation of text lines is performed by the directives, MOVE and DUP. MOVE copies and then deletes the specified lines from their original location, while DUP simply copies them.

As previously described, you can delete an individual text line by typing its line number, followed by a carriage return, but the DELETE directive allows the deletion of several lines, or line ranges at once.

Finally, there are two directives, MERGE and INSERT, which add text lines to EWFILE from another file. Lines added under INSERT must be inserted between two existing text lines, and are assigned line numbers by EDITOR. Lines added under MERGE are handled just as though they were entered from the terminal, and their line numbers may be either generated or taken from the text line itself. The directives described in this section manipulate text lines as a unit. To perform editing operations within the text line, see Intra-line Editing, Section 3.14.

3.13.1 Moving Text Lines — MOVE

The MOVE directive takes text lines from one position in EWFILE and inserts them in another part of the file. The lines at the original location are deleted.

MOVE,[$lnum_1$][,txt], TO $lnum_2$[,BY m][,AF][,CASE][,CRTL][,UNIT].

$lnum_1$ — the line numbers, or line number ranges (as defined in Section 3.2.2) of the text lines to be moved.

txt — a text search string. Only text lines containing the specified character string are moved. If $lnum_1$ is specified, the search is restricted to the indicated lines.

TO $lnum_2$ — the line numbers after which the specified text lines are to be inserted. If $lnum_2$ does not exist, lines are numbered starting from $lnum_2$, but if $lnum_2$ does exist lines are numbered starting from $lnum_2$ plus an increment defined by 'BY m'. Here, $lnum_2$ may be up to 20 line numbers (as defined in Section 3.4.1), resulting in up to 20 copies of the specified line range.

If ranges specified by $lnum_2$ overlap ranges specified by $lnum_1$, EDITOR issues a diagnostic saying that lines cannot be moved to a point within themselves. The directive is not executed.

BY m — a line number increment (as described in Section 3.4.7). If the specified value of m causes moved lines to overlap any existing text lines at the new location, EDITOR issues a diagnostic and does not execute the directive. EDITOR selects the largest possible increment not exceeding 10 if "BY m" is omitted.

AF — must be specified if CASE and CTRL are specified for string searches.

CASE — used in conjunction with the txt and AF parameters, specifies that the case of the search string must match (e.g. /Abc/ is not the same as /abc/). The default does not consider case (e.g. /Abc/ is the same as /abc/). CASE may be abbreviated C.

CTRL — if AF is also specified, CTRL prevents control codes from being ignored in a string search.

UNIT — specifies that all character strings must occur as a unit within the text line (as defined in Section 3.4.4). UNIT may be abbreviated U.

This directive deletes all text lines specified by $lnum_1$ and txt from their original position in EWFILE, and inserts them in the locations that directly follow the line numbers indicated by "TO $lnum_2$." If both $lnum_1$ and txt are omitted, EDITOR issues the diagnostic: *CANNOT MOVE ENTIRE FILE*. Moved lines are sequenced with the largest line number increment (an integer if possible) that will enable them to be inserted between the line number specified, and the next text line in the work file or by 10 if the largest integer is greater than 10. When text lines are moved to the end of EWFILE, they are given line numbers incremented by 10. Alternatively, by using 'BY m', you may specify increment values for the new line numbers to be generated. However, the specified values must not cause moved lines to overlap any existing text lines at the new location.

Examples:

NOTE: the examples assume the TO line number already exists.

1. Suppose EWFILE contains text lines numbered in increments of 10 from 10, 20, 30, etc., up to 500. Then typing:

```
MOVE,50-100,TO 200.
```

moves the six lines, 50 through 100, and assigns them the new line numbers: 201, 202, 203,...,206. Here the value 1 is the largest possible integer increment.

2. Given the same work file, typing:

```
MOVE,/FORMAT/ TO 5000,BY 10,UNIT.
```

moves all text lines containing the word FORMAT as a unit, to the end of EWFILE. The first such line encountered is renumbered as 5000, the second as 5010, the third as 5020, etc. Here, the parameter 'BY,10' is redundant since the value, 10, is the default increment when lines are moved to the end of EWFILE.

3. Again using the same work file, suppose you want to relocate a subroutine call which is currently somewhere between lines 100 through 200. Then typing:

```
MOVE,100-200,/CALL CONVERT/U TO 322,440.
```

instructs EDITOR to find the subroutine call and move it to two locations; 322 and 445. Here, the user-specified starting value applies only to the first location, and EDITOR selects the largest suitable integer increment (in this case, 5) for the second location.

3.13.2 Duplicating Text Lines — DUP

The DUP directive copies text lines from one position in EWFILE and inserts the copies in another part of the file. The original lines are unchanged.

DUP[,$lnum_1$][,txt], AT $lnum_2$[,BY m][,AF][,CASE][,CTRL][,UNIT][,VETO].

$lnum_1$ — the line numbers or line number ranges (as defined in Section 3.4.1) of the text lines to be duplicated.

txt — a text search string (as defined in Section 3.4.3). Only text lines containing the specified character string are duplicated. If $lnum_1$ is specified, the search is restricted to the indicated lines.

AT $lnum_2$ — the line numbers after which the specified text lines are to be duplicated. If $lnum_2$ does not exist, lines are numbered starting from $lnum_2$, but if $lnum_2$ does exist lines are numbered starting from $lnum_2$ plus an increment defined by 'BY m'. Here, $lnum_2$ may be up to 20 line numbers (as defined in Section 3.4.1). If ranges specified by $lnum_2$ overlap ranges specified by $lnum_1$, EDITOR issues a diagnostic saying that duplicating lines within themselves is illegal. The directive is not executed.

BY m — a line number increment (as defined in Section 3.4.7). If the specified value of m causes duplicated lines to overlap any existing text lines at the new location, EDITOR issues a diagnostic and does not execute the directive. If this parameter is omitted, EDITOR selects the largest possible increment not exceeding 10.

AF — allows you to specify CTRL and CASE for string searches.

CASE — used in conjunction with txt and AF parameters, specifies that the case of the search string must match (e.g. /Abc/ is not the same as /abc/). The default does not consider case (e.g. /Abc/ is the same as /abc/). CASE may be abbreviated C.

CTRL — if AF is also specified, CTRL allows you to search for strings containing control codes.

UNIT — specifies that the text search string must occur as a unit within the text line (as defined in Section 3.4.4). UNIT may be abbreviated U.

VETO — specifies that each line is to be displayed at the terminal before it is duplicated. You then enter one of several execution options (as described in Section 3.4.5). VETO may be abbreviated V.

This directive duplicates all text lines indicated by $lnum_1$ and txt at the locations that directly follow the line numbers indicated by 'AT $lnum_2$'. If $lnum_1$ and txt are omitted, the entire file is duplicated. The operation is identical to that of MOVE, except that the original lines are not deleted. Duplicated lines are sequenced with the largest line number increment (an integer if possible) that enables them to be inserted between the specified line number and the next text line in the work file or by 10 if the largest integer is greater than 10. When text lines are duplicated at the end of EWFILE, their new line numbers are incremented by 10. Alternatively, by using 'BY m', you may specify increment values for the new line numbers to be generated. However, the values specified must not cause duplicated lines to overlap any existing text lines at the new location.

Examples:

NOTE: the examples assume the AT line number already exists.

1. Suppose EWFILE contains text lines numbered in increments of 10 from 10, 20, 30, etc., up to 500. Then, typing:

   DUP 50-100 AT 200.

   will duplicate the six lines 50 through 100, and assign the new lines the numbers: 201, 202, etc., up to 206. Here the value 1 is the largest suitable integer increment.

2. Given the same work file, suppose you want to place a PRINT statement at several locations for debugging purposes. Then, typing:

   DUP /PRINT 500/U AT 100,140,200,BY 0.01,VETO.

   will duplicate the text lines containing PRINT 500 as a unit and will assign them the line numbers: 100.01, 140.01, and 200.01. If there are already two text lines containing PRINT 500, both will be duplicated at each location. However, this can be avoided by using VETO.

3.13.3
Deleting Text Lines — DELETE

The DELETE directive deletes the specified text lines.

DELETE[,lnum|@[abb]][,txt][,AF][,CASE][,CTRL][,UNIT][,VETO][,LIST]

lnum — the line numbers or line number ranges (as defined in Section 3.4.10) of the text lines to be deleted.

@abb — specifies a string abbreviation to be deleted. @ alone will cause the deletion of all abbreviations. See Section 3.14 for a description of string usage.

txt — a text search string (as defined in Section 3.4.3). Only text lines containing the specified character string are deleted. If lnum is specified, the search is restricted to the specified lines.

AF — allows you to specify CTRL and CASE for string searches.

CASE — used in conjunction with txt and AF parameters, specifies that the case of the search string must match (e.g. /Abc/ is not the same as /abc/). The default does not consider case (e.g. /Abc/ is the same as /abc/). CASE may be abbreviated C.

CTRL — if AF is also specified, CTRL allows you to search for strings containing control codes.

UNIT — specifies that the text search string must occur as a unit within the text line (as defined in Section 3.4.4). UNIT may be abbreviated U.

VETO — specifies that each line is to be displayed at the terminal before it is deleted. You then enter one of several execution options (as described in Section 3.4.5). VETO may be abbreviated V.

LIST — specifies that each line deleted is to be displayed at the terminal. LIST may be abbreviated L.

If lnum or @abb or txt is not specified and VETO is also omitted, EDITOR issues the message: *CANNOT DELETE ENTIRE FILE*.

Examples:

1. DELETE 88,88.2,90-97 VETO.

   The lines 88, 88.2, and 90 through 97 are displayed at the terminal. Each line for which the user types: Y is deleted from EWFILE. Typing N indicates that the line is to be retained.

2. Suppose a FORTRAN programmer wishes to delete a number of PRINT statements, which print debugging messages. Suppose, too, that this user had the foresight to confine these messages to a series of FORMAT statements having FORTRAN statement numbers: 5001, 5002, 5003, etc.

   DELETE /PRINT 500/(7,8).

   Note that the statements: PRINT 500 and PRINT 50000, are also deleted if they exist.

3.13.4
Resequencing Text Lines — RESEQ

The RESEQ directive renumbers text lines, but does not change their order within the work file. If no parameters are specified, all text lines are renumbered, such that the first line is assigned 100, and successive lines are assigned numbers incremented by 10. The optional parameters allow you to specify a line range to be renumbered, and the new line numbers to be assigned. However, the new line numbers may not cause any text lines being renumbered to overlap those not being renumbered, since this would constitute a re-ordering of the work file.

RESEQ[,lnum][,FROM n][,BY m][,FORMAT].

lnum — the line number range (as defined in Section 3.4.1) of the text lines to be renumbered. Only one line number range may be specified.

FROM n — the starting value for the new line numbers (as described in Section 3.4.6). To prevent overlap when lnum is specified, the value of n must be greater than the line number of the text line which immediately precedes the specified range, and less than the line number of the first text line following the range. If omitted when lnum is specified, the starting value is taken to be the first line number of the specified range.

BY m — the line number increment (as described in Section 3.4.7). If the user specifies a value for m which would cause renumbered text lines to overlap unrenumbered text lines, EDITOR issues a diagnostic and does not execute the directive. If omitted when lnum is specified, EDITOR selects a suitable increment (an integer when possible). The increment will never be greater than 10. When the specified line range includes the last line of EWFILE, the default increment is 10. Under SYSTEM,BASIC, m must be an integer value.

FORMAT — the EWFILE is resequenced but any established FORMAT boundaries are not altered. See Section 3.15.1 for a detailed discussion of FORMAT boundaries.

Examples:

1. RESEQ.

   This entry renumbers the entire work file. The first line is assigned 100; the second, 110; the third, 120; etc.

2. RESEQ,50-85.

   This entry renumbers the text lines between, and including, line 50 and line 85. The first line of this range is assigned 50, with successive line numbers incremented by a suitable value.

3. Suppose there are 100 text lines beginning with line 152 and ending with line 290. Then typing:

   RESEQ,152-290,FROM 200,BY 0.01

   will assign line 152 the new EDITOR line number, 200, and successive lines the numbers: 200.01, 200.02, etc., up to 200.99.

4. Given the same work file used in Example 3, typing:

RESEQ 152-290 FROM 100,BY 10.

would enter an invalid directive if: (a) there existed any text line in the range 100-151.999999, or (b) there existed any text line in the range 290.000001-990. In both cases, the directive would have caused new line numbers to overlap existing line numbers.

3.13.5 Inserting Text Lines from a File — INSERT

The INSERT directive inserts the entire contents of the file lfn into EWFILE following each of the lines specified by the parameter, AT lnum. When no other parameters are specified, the inserted text lines are assigned line numbers generated in increments of 10, with the starting value determined by the AT lnum parameter. Alternatively, you may instruct EDITOR to generate line numbers according to the starting and increment values specified by FROM n and BY m. In either case, if at any point in the insertion operation an assigned line number causes inserted text lines to overlap existing text lines, the line number increment is reset to 0.000001. If this measure proves insufficient, the increment is set to zero, and the remaining text lines are inserted with duplicate line numbers. You must then reinstate unique line numbers with the RESEQ directive. EDITOR issues an informative diagnostic whenever the line number increment is reset.

As under the directives OLD and MERGE, EDITOR truncates inserted text lines to the current LENGTH specification, while MARGIN, TAB, and SYSTEM have no effect.

INSERT,lfn,AT lnum[,FROM n][,BY m][,AF][,CASE][,CTRL][,NR][,VETO][,txt].

lfn — the name of a standard SCOPE coded file containing the lines to be inserted. If the lines begin with line numbers, those numbers will be included in the text. Appended line numbers, however, may be ignored.

AT lnum — the line numbers after which the new text lines are to be inserted. If lnum does not exist, lines are numbered starting from lnum, but if lnum does exist, lines are numbered starting from lnum plus a BY increment. Here, lnum may include up to 20 line numbers (as defined in Section 3.4.1).

FROM n — a starting line number (as defined in Section 3.4.6). The value of n must be greater than the first line number specified by 'AT lnum' and less than that of the next text line in the work file. If omitted, the starting value is computed from the line number specified by 'AT lnum' plus the increment value in effect. A user-specified starting value applies only to the first location indicated by 'AT lnum'.

BY m — a line number increment (as defined in Section 3.4.7). If the specified value of m causes inserted text lines to overlap any existing text lines, EDITOR automatically reduces its value and issues an informative diagnostic. If this parameter is omitted, generated line numbers are incremented by 10, 0.000001 or 0 depending upon the values of the existing line numbers. If 0.000001 is not small enough to insert all lines with unique line numbers, a 0 increment is used. EDITOR will issue a warning message in this case. The user can restore unique line numbers using the RESEQ directive.

AF — allows you to read an ASCII file. The default is to read the file in Display code. A warning message will be issued if the file appears to be the wrong character set. The warning turns VETO ON so you may correct the problem.

CASE used in conjunction with txt and AF parameters, specifies that the case of the search string must match (e.g. /Abc/ is not the same as /abc/). The default does not consider case (e.g. /Abc/ is the same as /abc/). CASE may be abbreviated C.

CTRL if AF is also specified, CTRL allows you to search for strings containing control codes.

NR the file lfn is not rewound before and after the operation. NOTE: if specified, the contents of the file lfn are inserted only at the first location specified.

VETO specifies that each text line is to be displayed at the terminal before it is inserted. You then enter one of several execution options (as described in Section 3.4.5). VETO may be abbreviated V.

txt only those lines matching the search string will be inserted into the work file.

The file lfn is read until end-of-information. All end-of-section and end-of-partition marks except those encountered at the end of the file are inserted as the text lines, *EOSnn and *EOP.

Examples:

NOTE: the examples assume the AT line numbers already exist.

1. Suppose you want to insert file A, which contains a 20 line subprogram, between lines 420 and 430 of the current EDITOR work file:

   INSERT,A,AT 420.

   The twenty lines are inserted into EWFILE and assigned the line numbers: 420.000001, 420.000002, etc., because the default increment of 10 proves to be too large for even the first text line inserted. Note that if EWFILE contains a FORTRAN program, file A must contain lines already in FORTRAN format.

2. Awkward line numbers that resulted in Example 1 could have been avoided by typing:

   INSERT,A,AT 420,BY 0.01.

   Here, the inserted text lines would be numbered: 420.01, 420.02, 420.03, up to 420.20.

3. To add the contents of file A to the end of EWFILE.

   INSERT,A,AT*L.

Character Set Warning

EDITOR checks the file lfn to make sure it appears to be in the correct character set. If it detects a line in the wrong character set, it issues a warning message like this:

WARNING—DISPLAY-CODE DATA EXPECTED ON FILE lfn MAY BE ASCII

Then EDITOR turns on VETO, allowing you to see the line in question, and decide whether to continue (C) or abort (S or K) the command. The warning will be issued only once per command.

A correctly formatted AF file will never be mistaken for Display code by EDITOR; however, certain rather unusual Display code text lines may look to EDITOR like AF, and trigger this warning erroneously. In this case, simply type "C" and EDITOR will continue. EDITOR will make this mistake on a Display code line that is 9 characters or longer, in which columns 1, 3, 5, 7 and 9 contain only the characters "A", "B" and "C".

3.13.6
Merging Lines from a File into EWFILE — MERGE

The MERGE directive enters text lines into a non-empty EWFILE from a standard coded file in the same manner as it enters text lines from the terminal. Existing lines may be replaced by lines with duplicate line numbers. MERGE inserts text lines according to their respective line numbers; line numbers are maintained in ascending order in EWFILE. The line numbers associated with the merged text lines may be generated by EDITOR, or taken from a column range within each text line. If only lfn is specified, the default processing is identical to that of the directive, OLD, where each line in file lfn must contain a valid line number in a column range determined by current settings of SYSTEM and LENGTH. Under SYSTEM,BASIC, these numbers must appear in columns 1-5; in any other system, they must appear at the end of each line, in the format created by SAVE.

If desired, you may specify the column range in which the line numbers begin, or you may instruct EDITOR to generate line numbers by specifying either (or both) 'FROM n' or 'BY m'. Generated line numbers are assigned sequentially in the order that the lines occur in lfn.

EDITOR truncates the merged lines to the current LENGTH specification, but MARGIN, TAB, and SYSTEM have no effect.

File lfn is read until end-of-information. However, end-of-section and end-of-partition marks are not inserted as text lines.

MERGE,lfn[,FROM n][,BY m][,AF][,CASE][,CTRL][,NR][,VETO][,txt].

lfn — the name of a local file in standard SCOPE coded format, which contains the lines to be merged into EWFILE.

FROM n — the starting value for line numbers generated by EDITOR (as described in Section 3.4.6). If both 'FROM n' and 'BY m' are omitted, line numbers are taken from the text. If only 'BY m' is specified, generated line numbers begin with 100.

BY m — an increment value for line numbers generated by EDITOR (as described in Section 3.4.7). If 'BY m' is omitted, but 'FROM n' is specified, the generated line numbers are incremented by 10.

AF — allows you to read an ASCII file. The default is to read the file in Display code. A warning will be issued if the file appears to be in the wrong character set. The warning turns VETO ON so the user may correct the problem.

CASE — used in conjunction with txt and AF parameters, specifies that the case of the search string must match (e.g. /Abc/ is not the same as /abc/). The default does not consider case (e.g. /Abc/ is the same as /abc/). CASE may be abbreviated C.

CTRL — if AF is also specified, CTRL allows you to search for strings containing control codes.

NR the file lfn is not rewound before and after the operation.

VETO specifies that each line is to be displayed at the terminal before it is merged. The user then enters one of several execution options (as described in Section 3.4.5). VETO may be abbreviated V.

txt only those lines matching the specified search string will be entered into the work file.

Examples:

1. The MERGE directive can be used to modify EWFILE by merging a separately constructed correction file.

First, you save or rename the file to be modified:

```
SAVE,A. SCRATCH.
```

Then, insert modifications into an existing EWFILE.

Another example similar to this, but less commonly used also comes after you have saved or renamed the file to be modified.

Enter correction text lines into a new EWFILE. When satisfied with the correction set, you might choose to insert an identifier name at the end of every text line by means of an intra-line editing directive (Section 3.14). Next, the correction set must be saved and the original file re-entered, so that the new text lines replace the old:

```
SAVE,MODS. OLD,A. MERGE,MODS.
```

In order to retain the original line numbers, take advantage of the default processing for OLD and MERGE, which uses the line numbers appended to each line by the SAVE operation. According to their line numbers, lines entered from MODS either replace, or are inserted between, the lines entered from A. Note that this technique can insert and replace text lines, but cannot delete them.

EDITOR work file MAIL is an alphabetized mailing list and file PHONE is an alphabetized list of telephone numbers. Each address on MAIL is one line long, each line in PHONE contains a name followed by a phone number.

MAIL=100-*L by 10, PHONE=100-*L by 10.

To combine use the directives:

```
USE,PHONE.SAVE,LIST.USE,MAIL.MERGE,LIST,FR 105,BY 10.
```

2. Both MERGE and INSERT allow you to combine files. In the directives below, MERGE adds the subprograms on files A, B, and C to the end of EWFILE, whose last line is numbered 500.

```
MERGE,A,FROM 510 BY 10.
MERGE,B,FROM 900 BY 1.
MERGE C FROM 1000.
```

The parameter, BY 10, is unnecessary in the first directive, since 10 is the default increment value. Thus, the lines from file A are numbered: 510, 520, 530, etc.; the lines from file B are numbered: 900, 901, 902, etc.; and the lines from file C are numbered: 1000, 1010, 1020, etc. In this particular example, one must be careful to specify starting and increment values which insure that subprograms do not overlap.

Character Set Warning

EDITOR checks file lfn to make sure it appears to be in the correct character set. If it detects a line in the wrong character set, it issues a warning message like this:

WARNING—DISPLAY-CODE DATA EXPECTED ON FILE lfn MAY BE ASCII

Then EDITOR turns on VETO, allowing you to see the line in question, and decide whether to continue (C) or abort (S or K) the command. The warning will be issued only once per command.

A correctly formatted AF file will never be mistaken for Display code by EDITOR; however, certain rather unusual Display code text lines may look to EDITOR like AF, and trigger this warning erroneously. In this case, simply type "C" and EDITOR will continue. EDITOR will make this mistake on a Display code line that is 9 characters or longer, in which columns 1, 3, 5, 7 and 9 contain only the characters "A", "B" and "C".

3.14 Intra-line Editing

Intra-line editing directives allow you to alter characters within a text line. The order of text lines within the EDITOR work file is unchanged. Intra-line editing involves editing operations performed on character strings.

A character string is a series of consecutive characters, which can be used in a variety of ways. A character string can be inserted in a text line. Its presence or absence can be used as a criteria for performing an operation on a given line. A character string can be used in its entirety or it may be represented by an abbreviation (see Section 3.16).

There are three editing operations which can be performed upon a line, namely:

1. replacement of one character string or column range by another;
2. insertion of a character string (either before or after a designated point within a line);
3. replacement of a character string or column range by blanks.

The general form of the intra-line editing directive is as follows:

txt_1 op txt_2 [txt_3][ALL][LIST][VETO][UNIT][[AF][CASE][CTRL]HOLD][lnum|@[abb]].

3.14.1
Basic Intra-line Editing

The essential part of the intra-line editing directive is: txt_1 op txt_2. In its simplest form: txt_1 indicates a character string or column range to be altered, op is an editing code for the operation to be performed, and txt_2 is a character string or column range which either replaces txt_1 or is inserted into the line at a designated point.

txt_1 specifies the column range or text string to be altered. txt_1 can be a simple character string only. Compound text strings are not allowed. This parameter has two forms:

1. (c_1,c_2)
which specifies a column range to be affected by the editing operation.

2. /chars/[(c_1,c_2)][U][N][C]

which specifies a character string to be affected by the editing operation. The optional subparameters modify the operation in the following fashion:

(c_1,c_2)	causes the editing operation to occur only if the character string begins within these columns.
U	causes the editing operation to occur only if the character string appears as a unit.
C	causes the editing operation to occur only if the case of the characters match. C is ignored unless AF in on.
N	causes the editing operation to occur whenever an attempt to find a match for the character string fails (including U and C checks).

txt_2 specifies the text string which replaces or is inserted at txt_1. txt_2 can be a simple character string only. txt_2 can be expressed in two forms:

1. (c_1,c_2)
which refers to the contents of the column range of the text line being edited. Those contents replace or are inserted at the point in the text indicated by txt_1.

2. /chars/[(c_1,c_2)][U][N][C]

The indicated character string replaces or is inserted after the text indicated by txt_1. The optional subparameters modify the operation in the following fashion:

(c_1,c_2)	causes the editing operation to occur only if the character string chars is found in the appropriate columns.
U	restricts the editing operation to lines in which the character string appears as a unit.
C	causes the editing operation to occur only if the case of the characters match. C is ignored unless AF is on.

N only lines which do **not** contain the character string chars are edited (including U and C checks).

Note: Using U, N and C with txt_2 is meaningless unless the column range (c1,c2) is also present (i.e., no string search for txt_2 is done unless a column range is given).

Examples of txt_1 and txt_2:

1. txt specified as a column range (c_1,c_2)

   (1,10)

   The first ten columns of every text line are used.

2. txt specified as a simple character string

   /PHONE/

   EWFILE is searched for lines containing the string PHONE.

3. txt appears as a simple character string with modifiers

   /PHONE/(1,10)U

   EWFILE is searched for lines which contain PHONE as a unit in columns 1 to 10.

4. txt specified as an ASCII string, requiring case matching.

   /August/C

   EWFILE is searched for lines containing the string, August, exactly as typed.

op a code for the editing operation to be performed. Legal codes are: =, I, L, B.

= **replacement**

Under replacement, the character string indicated by txt_1 is replaced by the character string indicated by txt_2.

Examples:

1. /FORMST/(7)=/FORMAT/

   replaces a misspelling beginning in column 7, with the correct spelling FORMAT.

2. /A1/=/R1/,30,31,400-420,ALL,VETO

   replaces all occurrences of the string, A1, in lines 30, 31, and 400 through 420, with R1. This directive might be entered to change FORTRAN formats from A1 to R1. Because UNIT is not specified, strings such as A10 or 5A1 are also modified, but the user can VETO any undesired changes when the text lines are displayed at the terminal.

3. (36-80)=/RESERVE 3 WORDS FOR EMPLOYEE NAME/,1480.

   changes the comment field of a COMPASS program statement at line 1480.

4. (1-5)=/200/,1310,H.

   change a FORTRAN line number at EDITOR line 1310.

I **(right) insertion**

With insertion, the character string indicated by txt_2 is inserted to the right of the character string indicated by txt_1.

Examples:

1. (72)I/JDATE2/,500-730

   inserts an identifier, JDATE2, into columns 73-78 of text lines 500 through 730.

2. /*/NI/ /,150,ALL.

   inserts a blank after every character (unless it is an asterisk) in line 150.

3. /CONTAIN/I/ING/,1470.

   changes the word CONTAIN to CONTAINING in line 1470.

L **left insertion**

With left insertion, the character string indicated by $txt_2$is inserted before the text indicated by txt_1.

Examples:

1. /408/(72)L/DATA/,610-760.

   inserts the identifier DATA before the string 408 when 408 begins in column 72 in text lines 610 through 760.

2. /./L/,PW=SECRET/,/ATTACH/(1),1-100,L.

   adds a password to any command to ATTACH any file in the line range 1-100.

B **blank**

The blank directive replaces the character string represented by txt_1 with blanks. txt_2 must be present, but is a dummy parameter, serving no function.

Examples:

1. /PHONE/B//

   searches for lines containing the string PHONE. The first occurrence of PHONE in each line is replaced by blanks.

2. /PHONE/(1,10)UB//

   searches for lines which contain PHONE as a unit in columns 1 to 10. The first such occurrence is replaced by blanks.

More Examples:

1. (36-80)=(21-80),1520-1640. (21-35)B//,1520-1640.

   moves text beginning in column 21 to column 36 for the line range 1520-1640. The same thing could be done with the command:

   (21)L/ ,1520-1640

   (if the blanks were carefully counted).

2. / /=////,120-650./ /I////,120-650.(1)L/ DATA /,120-650-

   .

   allows the user to enter FORTRAN DATA statements by entering just the variable names and values. The intra-line editing directives place slashes around the value and puts the keyword DATA before the variable names. For example:

   Before:

   120=A,B,C 1,2,3

   After:

   120= DATA A,B,C /1,2,3/

3.14.2 Additional Parameters

The remaining parameters refine the selection of lines to be edited, modify the form of the line after editing and control the processing and display of edited lines. These parameters give intra-line editing considerable power and sophistication.

Specifying lines to be examined for possible editing

These parameters define which group of lines will be searched when looking for text to be edited.

lnum lnum specifies a line number or line number range of text lines which may be edited.

@abb abb is the abbreviation which has been given to a character string via the STRING command (See Section 3.16). It indicates that the character string associated with abb is to be

edited. abb must be preceded by the @. If @ appears alone, all character strings which have been given an abbreviation may be edited. (See Section 3.16).

Note that an intra-line editing directive cannot operate on both text lines and strings with abbreviations simultaneously.

Specifying lines which will be edited

Within the group of lines which are searched, certain individual text lines, will be edited. The following parameters define the selection criteria.

txt_3 an additional character string to be searched for. While txt_3 is not an active member of the editing operation, it must be in any text line for editing to occur. txt_3 may be a compound character string.

UNIT specifies that each character string to be searched for must occur as a unit within the text line to be edited (i.e. txt_1, txt_2 and txt_3 character strings specified in the directive). UNIT may be abbreviated U.

Selecting the form of the line after editing

The user can determine how the editing operation will affect the text lines to be altered.

ALL specifies that the editing operation is to include all occurrences of txt_1. If ALL is not specified, only the first occurrence of txt_1 in the line is edited. ALL may be abbreviated A.

HOLD this parameter retains word positions after editing, words are kept in the same columns by removing or inserting blanks as required. This is useful when data has been entered using tab stops. HOLD may be abbreviated H.

Example:

```
SMITH          JOAN        31171
OLSEN          MARK        32619
FOGARTY        PATRICIA    51461
```

If HOLD is not used, the tabular alignment is lost when the following edit is performed:
/PATRICIA/=/PATRICK/

```
SMITH          JOAN        31171
OLSEN          MARK        32619
FOGARTY        PATRICK     51461
```

If HOLD had been used, the tabular alignment would have been retained
/PATRICIA/=/PATRICK/HOLD

```
SMITH          JOAN        31171
OLSEN          MARK        32619
FOGARTY        PATRICK     51461
```

HOLD has no effect on the B (blank replacement) editing operation, because tabular alignment is not altered.

If there are insufficient intervening blanks, forcing the columns to shift, EDITOR will issue the message "CAN'T HOLD COLUMNS" and turn on VETO. VETO will remain on unless you respond "C".

Controlling the processing and display of edited lines

As with other editing directives you can veto intra-line editing and produce a listing of all edited lines. See VETO (Section 3.4.5) and LIST (Section 3.8.1). VETO is automatically activated for any intra-line edit command which contains no line range (i.e. affects the entire work file).

Processing ASCII Files Using Intra-line Editing

AF causes the line to be edited in full ASCII. This means that lower case characters will not be folded to upper case before editing. Caution—Intra-line editing may fold lines at the wrong right margin or discard some ASCII characters if AF is omitted on an ASCII file. All control codes are discarded unless CTRL is specified.

CASE specifies that the case of the search string must match (e.g. /Abc/ is not the same as /abc/). The default does not consider case (e.g. /Abc/ is the same as /abc/). AF must be specified if CASE is to have any effect. CASE may be abbreviated C.

Note that the following two examples assume that "AF" is set ON.

Example:

/EDITOR/C=/Editor/

This directive replaces all occurrences of the word EDITOR (in upper case) with Editor (in mixed case).

CTRL if AF is also specified, all control codes will be included in editing. The default discards all control codes.

Example:

/Klingon vessel destroyed!!/I/[redacted],CTRL

This directive will cause the terminal bell to ring twice each time the message "Klingon vessel destroyed!!" is issued.

3.14.3 Continuation Lines

If intra-line editing causes a text line to extend beyond the current LENGTH specification, what happens to the excess characters depends upon the editing system being used.

BASIC, GENERAL or BATCH

Under these systems, the line may extend past the current maximum LENGTH, but not beyond column 140 without being truncated. Since the full line image will not be output under SAVE, LIST, or PUNCH. EDITOR issues a diagnostic, unless LENGTH is reset to prevent truncation.

FORTRAN

Under FORTRAN, a continuation line is created in accordance with FORTRAN rules; i.e., the continuation line contains a "+" (plus sign) in column 6, and, beginning in column 7, the text which overflowed the LENGTH limit. Up to four such continuation lines may be generated for each text line edited; each line is assigned a line number roughly halfway between the preceding and succeeding lines. At most the new line's number will be 10 greater than the preceding line's number.

Example:

Assume that LENGTH is set to 30 columns and that the work file, under SYSTEM FORTRAN, contains the following lines with the) in column 25.

```
550=10FORMAT (*HI THERE*)
560=PRINT 10
```

If the user types:

```
/THERE/UI/ SPORTS FANS/,550.
```

the resulting text lines will be:

```
550=10FORMAT (*HI THERE SPORTS
555=+ FANS*)
560=PRINT 10
```

TEXT

A TEXT continuation line is created by dividing the text line at the last blank character occurring before the LENGTH limit. If no blank characters satisfy this condition, the line is divided at the column specified by LENGTH. In either case, the remainder of the line begins in column 1 of a continuation line or that column set by MARGIN. The continuation line is assigned a line number roughly halfway between the preceding and succeeding lines. At most the new line's number will be 10 greater than the preceding line's number. Up to four such continuation lines may be generated for each text line edited.

Example:

Suppose LENGTH is set to 21 under SYSTEM TEXT, and that EWFILE contains the line:

```
10=THIS IS A SHORT LINE
20=
```

Here, the last non-blank character occurs in column 20. Consider the following two intra-line editing directives:

```
/A/I/ VERY/,10.
/A/I/ QUITE/,10.
```

The first produces:

```
10=THIS IS A VERY SHORT
15=LINE
20=
```

The second produces:

```
10=THIS IS A QUITE
15=SHORT LINE
20=
```

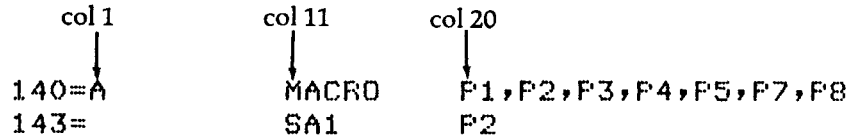
In the second case, the blank following SHORT occurs in column 22, which is beyond the specified LENGTH setting. Therefore, the line is divided between QUITE and SHORT.

COMPASS

A COMPASS continuation line is created in accordance with COMPASS rules; i.e., the continuation line contains a comma (,) in column one and, beginning in column two, the text which.overflowed the LENGTH limit. Up to four such continuation lines may be generated for each original text line edited; each is assigned a line number roughly halfway between the original text line and the succeeding line. At most the new line's number will be 10 greater than the original line's number.

Example:

Assume that length is set to 40 for this example and that the work file, under SYSTEM COMPASS, contains the following lines.



If the user types:

```
/P5/I/,P6/,140
```

the resulting text lines will be

```
140=A          MACRO     P1,P2,P3,P4,P5,P6,P7
141=,,P8
```

3.14.4 Folding Text Lines — FOLD

The FOLD directive causes all lines which are longer than the currently set LENGTH to be folded, that is, truncated at the column indicated by the LENGTH setting in a manner dictated by the current SYSTEM and the remainder of the line is continued on a new line immediately following the original line. Note that in SYSTEM GENERAL, SYSTEM BASIC or SYSTEM BATCH no line continuation is performed. FOLD, in this case, will not continue the line and issue an informative message to you for each line greater than LENGTH.

FOLD[,lnum][,txt][,AF][,CASE][,CTRL][,UNIT][,VETO][,LIST].

lnum specifies a line number or line number range of text lines which may be folded.

txt a character string which must be in any line for folding to occur.

AF causes the lines to be edited in full ASCII. Caution—FOLD may fail to fold some long lines, or discard some characters if AF is omitted on an ASCII file. All control codes are discarded unless CTRL and AF are specified.

CASE used in conjunction with the txt and AF parameters, specifies that the case of the search string must match (e.g. /Abc/ is not the same as /abc/). The default does not consider case (e.g. /Abc/ is the same as /abc/). CASE may be abbreviated C.

CTRL if AF is also specified, all control codes will be included in editing. The default discards all control codes.

UNIT all character strings specified in the directive must appear as a unit before folding will occur. UNIT may be abbreviated U.

VETO specifies that each line is to be displayed at the terminal before it is folded. You then enter one of several execution options (as described in Section 3.4.5). VETO may be abbreviated V.

LIST all folded lines will be listed at the terminal. LIST may be abbreviated L.

3.15 EWFILE Segmentation

It is often desirable to enter into an EDITOR work file more than one type of text line. For example, you may wish to construct a job consisting of a control section, a FORTRAN program, followed by some data. No single SYSTEM will accommodate the requirements of this application so you are allowed to define groups of text lines and to assign those groups the various attributes of SYSTEM, LENGTH, TAB, NOSEQ on SAVE, and HOLD on interactive editing.

When you are working in a segment of an EDITOR work file, the attributes of that segment are in force. For example, if the global system is BATCH and *JOBCARD* appears in columns 1-9 of a line in a BASIC segment, it will **not** be converted to a legal job card during a SAVE operation.

Up to seven different line formats may be specified. However, all of the lines of a specific format need not occur together. In fact, there may be any number of line groups associated with any specific format.

3.15.1 Defining Formats

The FORMAT directive defines a line format and defines the boundaries of that format within the work file. Up to seven unique formats may be defined for a work file. The following three formats are unique:

```
FORMAT COMPASS.

FORMAT GENERAL TAB 22 30 50.

FORMAT GENERAL TAB 12 35.
```

Any unique format may affect as many bounded areas of the work file as might be desired. To facilitate discussion, the parameter lnum is described last. However, lnum must appear **before** any TAB specification.

FORMAT,sysname[,lnum][,AF][,CASE][,CTRL][,LENGTH n][,TAB c_1...c_7][,NOSEQ][,HOLD]

sysname — any legal formatting system name allowed in the SYSTEM directive. This does not affect what action the GO directive might have, which is determined only by SYSTEM directive.

AF — causes EDITOR to behave in the AF mode for the given line range.

CASE — sets the CASE parameter ON for the given line range.

CTRL — sets the CTRL parameter ON for the given line range.

LENGTH n — sets the line length to n for all text lines associated with this format. If LENGTH n is not specified, then the current setting of the LENGTH directive will be in effect.

TAB c_1 ... c_2 — sets one to seven tab stops. c_1 through c_7 are column numbers at which a tab stop is to be recorded. The tab stops must appear in ascending order. Not specifying this parameter causes the TAB directive to take effect. If no tab stops are desired, TAB 0 (zero) should be specified.

It should be noted that when sysname is COMPASS, tab stops are automatically set at columns 11, 20, and 36 unless a different tab specification is given.

NOSEQ — causes all SAVE directives to be performed with the NOSEQ parameter, unless a SEQ parameter is given on the SAVE directive. This parameter does not affect any directive other than SAVE, or directives which initiate SAVE, such as FTN or BATCH. NOSEQ may be abbreviated NS.

HOLD — causes all intra-line editing directives to be performed with the HOLD parameter unless an NHOLD parameter is specified on the intra-line editing directive. HOLD may be abbreviated H.

lnum — a list of one to twenty line numbers or line ranges representing the boundaries of the format. When no line numbers are specified, the format will affect the entire EDITOR work file.

A single line number causes a boundary to be recorded at that line number. The specified format takes effect at that line and affects all lines following it up to the next recorded format boundary in the work file. If no other format boundaries occur following that line, then the entire EWFILE from that line on will be affected.

A line range causes a boundary to be recorded at the first line number as above, but at the second line number plus .000001 another boundary is recorded in the work file to reinstate the previously existing format. If no format existed, EDITOR uses the current global SYSTEM, LENGTH and TAB for this format.

Examples:

1. Consider the following directive sequence:

```
SYSTEM GENERAL. LENGTH 80. TAB 30.

FORMAT FORTRAN 100-1000.

FORMAT TEXT 700-800X, LENGTH 72
```

produces the following segmentation of the work file:

lines	attributes
*F-99.999999	current global attributes
100-699.999999	FORTRAN
700-799.999999	TEXT, LENGTH 72
800-1000	FORTRAN
1000.000001-*L	current global attributes

In this example, two format types are defined: (1) FORTRAN and (2) TEXT with LENGTH=72.

Since a maximum of seven different format types may be defined, up to five more might be defined. Note that this does not limit the number of segments. For example:

```
FORMAT GENERAL 1000X-2000, 5000X-7000, 8000X-*L, TAB 16, HOLD.
FORMAT BATCH *F-100X, NOSEQ.
FORMAT COMPASS 500-700X.
FORMAT GENERAL 2000X-3000, NOSEQ, LENGTH 90.
FORMAT BASIC 3000X-5000, 7000X-8000.
```

This would result in the following format boundaries:

lines	attributes
*F to 99.999999	BATCH with NOSEQ
100-499.999999	FORTRAN
500-699.999999	COMPASS with TAB 11, 20, 36
700-799.999999	TEXT with LENGTH=80
800-1000	FORTRAN
1000.000001 - 2000	GENERAL with TAB 16 and HOLD
2000.000001 - 3000	GENERAL with LENGTH=90, NOSEQ
3000.000001 - 5000	BASIC
5000.000001 - 7000	GENERAL
7000.000001 - 8000	BASIC
8000.000001 - *L	GENERAL

2. This example illustrates the use of AF mode in a file containing both ASCII and Display code information.

```
SET AF OFF.SYSTEM,GENERAL,SETFILE.
FORMAT,GENERAL,1-100,LENGTH,80,NS.
FORMAT,TEXT,100,AF,LENGTH,120.
```

This allows an ASCII text record starting at line 100, with a Display code exec file beginning at line 1 which is executed when you type "GO". If there is a *EOS (or *EOR) just before the ASCII text, SETFILE will contain a Display code record followed by an AF record. The SETCODE for the file will be OFF, since it isn't all ASCII.

3.15.2
Listing Formats

Typing the FORMAT directive with no parameters produces a listing of all current format boundaries in effect in the work file. If a format boundary was set at *F (first line of the work file) then the listing will locate it at line 0.

FORMAT.

3.15.3
Clearing Formats

All formats may be cleared by using the FORMAT directive with no line numbers or line number ranges, and specifying the same editing system as used in the last SYSTEM command.

It is advisable to use the EDSTAT directive prior to attempting to remove all formats to insure that the proper current global attributes are listed on the FORMAT directive.

For example, if EDSTAT indicated the current system as GENERAL, and the tab stops as 30,45 then you could clear all the formats by typing:

FORMAT, GENERAL, TAB 30,45. or FORMAT,GENERAL.

The message "ALL FORMATS DELETED" would then be sent to your terminal.

3.15.4
Resequencing an EDITOR Work File Containing Formats

The RESEQ directive can be used to resequence the work file without moving format boundaries. To accomplish this, the FORMAT parameter must appear on the RESEQ directive. RESEQ will attempt to resequence each work file segment between format boundaries separately. The example below should help illustrate this.

The following format is set:

```
FORMAT, FORTRAN *F-1000X.
FORMAT, GENERAL, TAB 10, 1000-*L.
```

The following lines are entered:

```
100=PROGRAM XYZ (INPUT,OUTPUT)
110=C COMMENT CARD
120=READ 100,IX
.
.
.
1000=100                    10
1010=110                    12
1020=145                    15
```

You then resequence the file by typing:

```
RESEQ, FROM 10 BY 1, FORMAT.
```

The work file now contains the following lines:

```
  10=                          PROGRAM XYZ (INPUT,OUTPUT)
  11=C                       COMMENT CARD
  12=                          READ 100, IX
  .
  .
  .
1000=100                       10
1001=110                       12
1002=145                       15
```

3.16 Abbreviations for Character Strings

EDITOR provides you with the ability to define abbreviations for character strings. You can refer to an abbreviation for a character string. Whenever EDITOR encounters the abbreviation, it will be interpreted before the directive is processed.

3.16.1 Defining Abbreviations for Character Strings

The STRING directive defines a name to be used as an abbreviation for a string of characters. Whenever the abbreviation (delimited by double quotes) is found in an EDITOR directive, the abbreviated name is replaced by the string itself.

```
STRING,@abb,/[chars]/
```

@abb — the abbreviation for the character string, which consists of 1 to 6 alphanumeric characters prefaced by an at-sign (@). Note that the first character of the string abbreviation must be a letter.

/chars/ — the character string chars delimited by slashes (/) may be from 0 to 140 characters in length. A single slash (/) is represented in the character string by two adjacent slashes. Omitting the chars parameter causes a null string to be associated with the abbreviation. Nested string definitions are not allowed.

Examples:

1. `STRING,@A,/FULL,100-500/.`

   defines the string A as the character string, FULL, 100-500.

2. `STRING,@XX,///ABORT//=//TERMINATE//,ALL/.`

   defines XX as the character string, /ABORT/=/TERMINATE/,ALL.

3. `STRING,@NULL,//.`

   defines NULL as a null string.

3.16.2
Using Abbreviations for Character Strings

Once an abbreviation has been defined then the string abbreviation delimited by double quotes (") may be entered on any EDITOR directive.

Examples:

1. Given that the string named A is associated with the character string, FULL, 100-500. Then typing:

   LIST,"A".

   will cause the following directive to be executed:

   LIST,FULL,100-500.

2. A string named XX is defined as the characters, /ABORT/=/TERMINATE/,ALL. Then, typing:

   -"XX",100,300,550.[1]

   expands to:

   -/ABORT/=/TERMINATE/,ALL,100,300,500.

3.16.3
Listing, Deleting and Saving Strings which have Abbreviations

Five EDITOR directives can operate on strings which have abbreviations; outputting, deleting or editing. These are the LIST, LISTF, SAVE, DELETE and intra-line editing directives. Referencing a character string is similar to referencing a line number or line range, with the restriction that no directive may operate on character strings and text lines simultaneously.

When used with these directives, the string abbreviation must be prefaced by an @. If the @ symbol is not immediately followed by an abbreviation name, it refers to all character strings with currently defined abbreviations. For a full discussion of directive syntax, see the relevant section in this chapter on each directive. The following examples show directives using abbreviations for character strings.

Examples:

1. To list all abbreviations with their assocated character strings, type:

   LIST,@.

   Continuing the example used in Section 3.16.2, this would produce the output:

   A=FULL,100-500
   XX=/ABORT/=/TERMINATE/,ALL

---

[1]Note that the minus sign is needed to ensure that the directive is interpreted as an EDITOR directive (see Section 2.1.4).

2. To delete the abbreviation XX, type:

   DELETE,@XX.

3. To save all abbreviations with their associated character strings on a file so they might be later re-entered via a READ directive, type:

   SAVE,SFILE,SOURCE,@.

   Note that with the SAVE directive, abbreviations are illegal without the SOURCE parameter.

4. To alter the character string associated with a string name without completely replacing it, the following might be used:

   /FULL/L/NS,/,@A

   This would insert "NS," to the left of the characters "FULL" in the character string for which A is the abbreviation.

3.17
EDITOR Work File Status — EDSTAT

The EDSTAT directive lists the current status of work file attributes. EDSTAT is automatically executed when you log in if there is an existing retained EWFILE. Below is a sample output with the maximum information that may be present. Items or entire lines which are not applicable are omitted by EDSTAT.

EDSTAT.

Sample:

```
[1]PF, COM[2]PASS([3]UP), 57 [4]LINES 100-[5]670 ([6]FMT) LENG[7]TH 72 USENA[8]ME=EW
MARG[9]IN 2, TAB 11[10] 21 36 ([11];), GOFI[12]LE=EXX
LOCK[13]=PART NR[14]=U=NS=F=A=L=UP=H=FMT=SO=ON STRIN[15]G=OFF 3 STRI[16]NGS
```

NOTES:

1. present if the EWFILE is a permanent file
2. the current system as defined by SYSTEM
3. (UP) is present if the UPDATE parameter was specified on the SYSTEM directive
4. the number of lines in the current work file
5. the range of the line numbers in the EWFILE

6. present to indicate any segmented FORMAT(s)

7. the current global line length. (Line length may vary from one segment to another.)

8. the original file name of the EWFILE that appeared on the USE directive

9. the left margin setting

10. currently set tab stop columns

11. the tabulation character as set by TABCH

12. the file name to be EXECed as supplied on the SYSTEM directive

13. the status of the EWFLOCK

14. list of options set on by the SET directive (see Section 3.19 for the full option names)

15. present if SET STRING=OFF was specified

16. number of current strings defined by the STRING directive.

An empty EWFILE with no attributes set would produce the following EDSTAT output:

```
FORTRAN, NO LINES, LENGTH 72
```

This message displays the default editing system — FORTRAN, the number of lines in the file, and the default line length — 72 characters.

3.18 Locking the Work File — EWFLOCK

EWFLOCK may be used to protect the EWFILE from accidental alteration. When EWFLOCK ON is specified the EDITOR will abort any directive which would change the contents of the EWFILE. The selection of PART will cause EDITOR to select VETO automatically in any directive which might change the work file, or, in the case of a directive which does not allow VETO, EDITOR will ask you for a response (Y or N) before continuing.

EWFLOCK OFF defeats the above safeguards. This is the default.

```
EWFLOCK [ON|OFF|PART].
```

3.19 Changing Default Conditions — SET

This directive allows you to alter the default condition for several EDITOR parameters. The default condition may be set on or off for each parameter or its negative.

```
SET,param=ON
or
SET,param=OFF
```

For example, the normal default for the UNIT parameter is OFF, that is, UNIT must be specified whenever it is desired.

SET UNIT=ON will cause the UNIT parameter to be assumed unless NUNIT (the negative of UNIT) is specified.

Following is a list of the parameters that may appear on a SET directive. The legal abbreviations are underlined with the negative mnemonic in parenthesis. These parameters are discussed in Appendix G and with the directives with which they are used.

parameter	usage
AF(NAF)	process the file using the ASCII character set
ALL(NALL)	edit all occurrences of a string in a line
CASE(NCASE)	perform case matching with search strings
CTRL(NCTRL)	copy control codes
FORMAT(NFORMAT)	resequence within existing segments
FULL(NFULL)	do not compress blanks on output
HOLD(NHOLD)	hold columns during editing
LIST(NLIST)	list in full all edited lines
NOSEQ(SEQ)	do not output sequence numbers
NR(R)	do not rewind a file
SOURCE(NSOURCE)	output the source image
STRING(NSTRING)	allow string substitution (this is normally ON)
UNIT(NUNIT)	perform unit checking with search strings
UPDATE(NUPDATE)	select UPDATE mode
VETO(NVETO)	prompt for a veto response

Setting the positive form ON is equivalent to setting the negative form OFF, as given in parentheses in the list above. Thus, SET UNIT=ON is equivalent to SET NUNIT=OFF.

3.20 UPDATE and EDITOR

UPDATE is a CDC utility used to maintain large card image files (normally programs).[2] It has line-by-line editing capabilities but no intra-line editing facilities. EDITOR can be used to construct UPDATE correction sets for a given portion of an UPDATE program library.

It is assumed here that you are familiar with UPDATE and that you have previously created a program library. You should also note that EDITOR does not recognize COMDECKs.

The first step is to create a COMPILE file copy of the UPDATE decks to be modified. This could be done as follows:

```
OK-PROMPT.  CONNECT,INPUT.
OK-UPDATE,Q,L=0.
**COMPILE,deck names
**EOS
OK-
```

The UPDATE command should not contain the parameters 8 or D as this will cause a COMPILE file to be generated with sequence information that is incomplete. Once the COMPILE file has been generated, it is given to EDITOR via the OLD command, for example:

```
OLD, COMPILE, UPDATE, FROM 100 BY 10.
```

[2]UPDATE is described in full in the *Control Data Corporation UPDATE Reference Manual.*

Note that the UPDATE sequencing is completely separate and distinct from the EDITOR line numbers. Thus, the FROM n or BY m parameter should be used when the UPDATE parameter on OLD is used.

Following the OLD directive, the EWFILE will contain the contents of the UPDATE COMPILE file. You may then proceed with any editing desired. When such editing is complete, several options are available to you:

1. If the text in the file is a complete program, subroutine, etc., it can be compiled directly just as any normal EWFILE might be by using the appropriate compilation directive.

2. An UPDATE correction set may be constructed by EDITOR. Before this step is taken, the user must insert before the first line of the COMPILE file in the work file, an UPDATE *IDENT card and any other UPDATE directives necessary, as EDITOR will not supply these by itself. Then, to create the UPDATE correction set, you must type:

   SAVE, lfn, UPDATE.

This will cause a correction set to be written on the file lfn, which may then be used to modify a current UPDATE program library.

Also note that if you have a *WEOR UPDATE directive in an EDITOR work file, and do a regular SAVE, that UPDATE will scan as far as column 80 looking for a level number after the *WEOR and end up using the EDITOR line number as the level number. UPDATE may write an end-of-partition (EOP) for an *WEOR without a level number since the EDITOR line number will usually be larger than 17. Similarly, UPDATE directives that include an optional local file may attempt to use an EDITOR line number as the file name.

Entering UPDATE Correction Sets

Another way that EDITOR may be used in conjunction with UPDATE is to construct an EWFILE consisting of an UPDATE correction set. You may use any SYSTEM other than BASIC without having to worry about EDITOR reformatting UPDATE directives (the * in column one of such directives inhibits reformatting).

Once you have constructed the correction set in the work file, then you may use compilation directives (e.g. FTNX) to call UPDATE and compile the program. Before typing the compilation directive, however, you must type a SYSTEM directive with the UPDATE parameter specified. With that done, any compilation directive will cause UPDATE to be called before the compiler requested is executed. (See Section 3.10 for details.)

3.21
Using EDITOR to Process ASCII Fancy Files

Internally, the contents of EWFILE are recorded using an upper/lower case character set. Even while operating in DC mode, EDITOR records input lines in upper/lower case. Lower-case characters are folded to upper case during DC editing and processing. AF mode is characterized by input, output and editing of text using the full ASCII character set. Control characters may be ignored or processed at the user's option.

All input/output directives allow you to process ASCII files, as do all editing directives. You can specify ASCII processing by using the AF parameter on these directives or by using SET or FORMAT to establish AF mode as the default. Additional parameters allow you to control the processing of case and control characters during string searches.

3.21.1 The ASCII Fancy Parameters

The following parameters (AF, CTRL and CASE) are used only when preparing and processing ASCII Fancy files. They are used in conjunction with one another. The AF parameter enables EDITOR to produce ASCII output files. CASE and CTRL are meaningless if AF is not specified. If you are processing ASCII information, we recommend that you specify 'SET,AF=ON' (Section 3.19) or specify AF on a FORMAT directive (Section 3.15.1). This makes ASCII Fancy processing the default mode for the file or the specified line range (respectively).

AF enables the AF mode of EDITOR processing. It is necessary on the OLD, INSERT and MERGE directives if the input file is ASCII. On the SAVE and LISTF directives, it causes the output file to be in the ASCII character set. On the LIST directive, AF causes the contents of EWFILE to be displayed in upper/lower case.

CTRL prevents the suppression of control characters on string searches and the output of the SAVE, LIST and LISTF directives. If you require control characters in the text, we recommend using 'SET,CTRL=ON'. (Note: Control characters are never suppressed on input to EWFILE; only on output.) Caution: Both intra-line editing (Section 3.14) and FOLD (Section 3.14.4) directives will remove control characters from any lines they change, unless CTRL is on.

CASE On any string search, both the string and the line to be searched are normally folded to upper case before searching. So, the case of the search string characters is not normally considered when searching for a match. If you wish to consider the case of the characters in the search string, specify the CASE parameter on the appropriate directive. CASE is a legal parameter on any directive which allows string searching.

This function may also be performed by concatenating a 'C' to an individual search string (e.g. '/Option/UC').

3.21.2 ASCII String Matching

As mentioned in the discussion of the CASE parameter in Section 3.21.1, lines are usually folded to upper case before performing a string search. This folding before searching does not alter the content of EWFILE. Specifying AF and CTRL suppresses the deletion of control characters before matching strings. This allows you to search for control codes in EWFILE. However, you must be aware of some possible pitfalls in using EDITOR with both AF and CTRL on. Consider the following text lines:

```
100=MY[DC1]STRING
110=MY[HT][HT]STRING
```

1. Control codes will separate alphabetic strings for the purposes of UNIT checking. /STRING/ is a unit in both lines; /MYSTRING/ is not.

2. EDITOR counts all characters, including control characters when computing the length of a line. Each ASCII code counts as one character in the line, regardless of its value. Therefore, line 100 is 9 characters long, and line 110 is 10.

3. If graphic interpretation of control characters has not been specified using the %SHOWNPC command (Section 8.3.11), control codes entered by mistake or because of garbled transmission will not be visible, and may cause very confusing EDITOR behavior.

3.21.3
Intermixed ASCII and Display Code

You may set up an EDITOR work file which contains both AF and DC data. For example, the file might be constructed this way:

```
SYSTEM,BASIC,SETFILE.
FORMAT,GENERAL,10.
FORMAT,FORTRAN,100.
FORMAT,TEXT,AF,1000.

10=SKIPF,SETFILE.
20=FTN,I=SETFILE.
30=LGO,SETFILE.
40=*EOS
100=     PROGRAM LETTER (INPUT,OUTPUT)
    .
    .    a program which formats text into a
    .    standard business letter style.
45=      END
460=*EOS
1000=TO:      John Barker
    .
    .
    .
FROM:   Pat Verbrick
    .
    .
    .
```

When the user types 'GO', the exec file at the beginning of the work file will compile and execute the FORTRAN program. The exec section and the FORTRAN program must be translated into Display code while the input data is in ASCII.

Note that AF is not SET on for the work file. This allows the exec and program section to remain in DC mode while the rest of the work file is processed in AF mode because of the AF specification on the FORMAT directive. Any editing done on the exec and program sections will be done in DC mode, while the data after line 1000 will be edited as full ASCII.

Although the *EOS lines are in lower case, they will still generate SCOPE end-of-section marks on SETFILE. The special processing for *EOS (or *EOR), *EOP (or *EOF) and *JOBCARD* in column 1 of a text line will take place without regard to the character case.

Because the file written by the SAVE directive contains segments in more than one character set, the familiar sequence:

```
SAVE,X.
SCRATCH.
OLD,X.
```

or equivalent operations will not produce the desired result. The OLD command can deal with only one character set in a file. DC and AF data in the same work file must come from different source files.

The sequence of commands:

```
SAVE,X,SOURCE.
SCRATCH.
READ,X.
```

will work, however, because the READ command can use files written in both character sets. This works because the SOURCE parameter puts the line numbers at the beginning of the line, which allows EDITOR to automatically determine the character set of the line.

4

Chapter 4 is reserved for future descriptions of software packages for the interactive system.

5

Interactive I/O—Commands and Language Facilities

Interactive users of the main computer system often communicate with executing programs. The program can ask you for input, you can respond, and the program can take differing actions based on your response. Such communication occurs through the use of connected files. Programs that need user-typed responses can issue read requests on a connected file; programs transferring data to a terminal issue write requests 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. NOTE: Only temporary (local) files may be connected. Permanent files cannot be connected.

All file-positioning commands (e.g., SKIPF and REWIND) are ignored when made on connected files.

It is important to distinguish between connected file input/output and communication with the system itself. System commands are not read by means of connected files. When you call a program by typing a command (e.g., FTN,HAL,LGO,RFL) the system reads the command directly. However, some system programs, such as AUTHORF, will connect files for I/O once they have been called. To make this distinction clear, suppose you are prompted by the OK- message. The terminal job is said to be "Waiting command." Then, if you type AUTHORF, the system program AUTHORF will connect the files INPUT and OUTPUT. AUTHORF will then write "INPUT..." on the file OUTPUT, causing the message to show at your terminal. The program will then issue a read request on the file INPUT. Until you respond, the terminal job is said to be "Waiting input."

5.1
Character Sets

A character set is a collection of graphics (letters, digits, and special symbols) which the computer recognizes. Each graphic of a character set is associated with a number called a 'code', which represents the character within the computer.

There are five character sets available on the interactive system. Connected files can be written in DC (Display Code), AS (ASCII), AF (ASCII Fancy) BI (Binary), and BF (Binary Fancy). The following table outlines the differences between the interactive character sets.

Character Set	Graphic Set	Character Representation	End-of-Line Character	Carriage Control
DC (Display Code)	63-character ASCII graphic subset	6 bits per character packed 10 per central memory (CM) word	0 in bits 0-11 of a CM word	first character of each line
AS (ASCII)	128-character full ASCII graphics	7 bits per character packed 5 per CM word	12-66 bits of zero in a CM word	none
AF (ASCII Fancy)	128-character full ASCII graphics	7 bits per character packed 5 per CM word	0 in bits 0-11 of a CM word	first character of each line
BI (Binary)	none	8 bits per character packed 5 per CM word no parity bit	none	none
BF (Binary Fancy)	none	8 bits per character packed 5 per CM word no parity bit	12-66 bits of zero in a CM word	first character of each line

Terminals transmit ASCII (American 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 the device control characters described in Chapter 8.

The DC character set is a 63-character ASCII graphic subset. It does not contain lower case letters and lacks some of the special symbols available in full ASCII. The DC character set is identical to the CDC Display Code routinely used by programmers. This allows you to utilize interactive debugging aids in the development of programs for batch use. An interactive system user who chooses the DC character set causes the system to convert the full ASCII characters sent by the terminal into the internal representation of Display Code for the mainframe.

Interactive system users who wish to use the full ASCII character set have the option to do so with AS and AF connected files. This is used in most text processing applications where upper and lower case are desirable.

Binary files are used for the transmission of 8-bit binary information from minicomputers and for the processing of 8-level paper tapes.

5.1.1 DC (Display Code) Files[1]

If a file is connected Display Code (DC), information is read from the terminal and transferred to the program doing the read in Display Code.

When a program reads from an DC connected file, the system translates the 7-bit full ASCII characters typed by you into 6-bit long CDC Display Code characters. These characters are packed 10 per word in the buffer of your program. The system maps each character from the full ASCII character set to an appropriate Display Code equivalent. Upper case letters are mapped to

[1] DC (Display Code) is equivalent to OM (Old Mistic). The OM character code may be used where ever a reference to DC is made.

the equivalent Display Code value, as are numerals; lower case letters are mapped to the corresponding upper case equivalents. All special characters (except %) are mapped from full ASCII to Display Code.

Control characters cause their various functions to occur regardless of the fact that DC input is occurring.

DC connected files are similar to files destined for printing on line printers in that each output record begins with a carriage control character. Depending on the character beginning the unit record (or "line"), the system sends the ASCII code for the terminal carriage return (CR) or line feed (LF) in appropriate combinations.

Note that the carriage control character is not printed (unless (%CCTL is OFF). DC and ASCII Fancy carriage control and the corresponding ASCII codes sent to the terminal are as follows:

1	CR,3LF
0	CR,2LF
+	CR
-	CR,3LF
,	no action
other	CR,LF

Thus, if a DC output line begins with the character +, the carriage will be positioned to the leftmost point, and the rest of the record will print over the last line output. A blank carriage control will result in output beginning on the next line after a carriage return.

Unit records of DC connected files are terminated by "zero bytes." A zero byte is defined as 12 bits of binary zero at the end of a word. Each unit record of a DC file will end with a zero byte, plus as many zero bits as are needed to pad out the word. Thus, the last word of a DC record will have 12, 18, 24, 30, 36, 42, 48, 54, 60 or 66 zero bits—depending on the number of Display Code characters in the word. (Users familiar with Cyber Record Manager will note that DC files correspond to files of Record Type Z, Blocking Type C.)

On input, the zero byte is sent to the program to signify end-of-line. As an example, assume you enter

A DC INPUT.

in response to a read operation on a connected DC file. The system would place in the program's buffer:

WORD 1	A	b	D	C	b	I	N	P	U	T
octal	01	55	04	03	55	11	16	20	25	24
binary	000001	101101	000100	000011	101101	001001	001110	010000	010101	010100

WORD2							zero byte indicating end-of-line			
octal	57	00	00	00	00	00	00	00	00	00
binary	101111	000000	000000	000000	000000	000000	000000	000000	000000	000000

Note how the system padded the last (in this case the second) word of input with binary zeroes to indicate end-of-line.

When the system is processing write operations on DC files, the zero byte is taken as an end-of-line marker. Any trailing blanks in the line to be written are not sent to the terminal. Any binary zero characters within a word (not part of the end-of-line byte) are converted to blanks and sent to the terminal with the line. The next character after the end-of-line is used as the carriage control for the next output line.

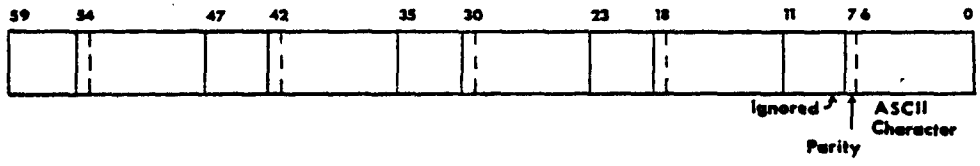
Important: Note that high-level languages, such as FORTRAN, process connected operations on DC files as if from batch. For instance, formatted FORTRAN read operations from a terminal are analogous to read operations from the batch input stream. Users of high-level languages for DC-processing need not worry about such considerations as end-of-line bytes; the compiler does that work. Programs in any high-level language that are normally run from batch can therefore be converted for interactive use with minimal effort. (See Section 5.8 on language facilities.)

In general, COMPASS programmers will specify recall for connected I/O to ensure the appropriate order of operations. Completion of a write operation to a connected file does not imply that data has, in fact, finished printing at the user's terminal; rather, completion of a write operation merely implies that the system has read all of the information from the user program's output buffer.

5.1.2
AS (ASCII) Files

AS files provide terminal/program communication in the ASCII character set. Characters other than control characters are sent to programs on input in the ASCII character set. On output to an AS file, the characters are sent to the terminal verbatim (no translation or mapping is done).

ASCII files at MSU contain words with 5 12-bit parcels, the ASCII byte being in the lower 7 bits of each 12-bit parcel.



On input from an AS file, end-of-line is indicated by a parcel of 4000_8 (octal). This "end-of-line parcel" is the only character parcel with a non-zero bit in the upper 5 bits of the parcel. As many 4000_8 parcels as needed are used to pad the last data word. If an exact multiple of 5 characters is read, the last word placed in your buffer for the line will consist of 5 parcels of 4000_8.

Note that if 4000_8 is output as a character on an AS file, nothing will be sent to the terminal for that character. Thus, if you wish to output a line read in from an AS file you must insert your own carriage return (CR) where the 4000_8 parcel exists.

You should also note that some terminals must wait for the carriage to return before they can resume printing when a CR is received. Users of AS files should take this into account when writing information to a terminal. You may find it necessary to insert ASCII nulls after outputting carriage returns. (See Section 8.1.2)

Unlike DC files, AS files never have carriage controls. If you are writing on an AS file you must supply the desired combination of carriage returns and line feeds.

5.1.3
AF (ASCII Fancy) Files

AF (ASCII Fancy) files are connected files designed to take advantage of the entire ASCII character set while allowing for use of DC style carriage controls. As is the case for AS files, the 7-bit character is packed in 12 bit parcels, 5 to a central memory word. As in DC files, the end-of-line causes 12 to 60 bits of binary zero to pad the data word. As is the case for DC files, on output the character following each zero byte is taken as a carriage control. Note that the carriage control character is never printed (unless %CCTL is OFF.) AF carriage controls have these effects:

1	CR,3LF
0	CR,2LF
+	CR
-	CR,3LF
,	no action
other	CR,LF

If you typed:

```
ASCIIFILE
```

followed by a carriage return in response to a read on a connected ASCII Fancy file, the program would read values packed as follows:

WORD 1	A	S	C	I	I
octal	0 1 0 1	0 1 2 3	0 1 0 3	0 1 1 1	0 1 1 1
binary	000001000001	000001010011	000001000011	000001001001	000001001001

WORD2	F	I	L	E	end-of-line
octal	0 1 0 6	0 1 1 1	0 1 1 4	0 1 0 5	0 0 0 0
binary	000001000110	000001001001	000001001100	000001000101	000000000000

Because a byte of binary zeroes is reserved for the end-of-line byte, the character NUL, whose value is 000_8, is represented as a 4000_8 in AF processing.

You can choose to automatically output CRs and LFs to AF files at will. The necessary number of delay characters will be sent to the terminal. An imbedded CR on output is not interpreted as an end-of-line: the character after a CR or CR/LF is never taken as a carriage control and is always sent to the terminal.

5.1.4
BI (Binary) Files

BI files are essentially AS files composed of 8-bit characters. Like AS files, 5 parcels of character data are packed in each CM word; for BI files, however, the parity bit is transmitted as data. End-of-line is represented by a 4000_8 parcel or by a word of 4000_8. On output, the system will not insert delay characters after CR of LF on BI files. BI files must have the CR/LF codes the user desires for proper carriage movement: there is no end-of-line character. If more than the current maximum of input characters is entered (See Section 8.3.9, INLEN) a carriage return is not inserted.

Reading of BI files (with the aid of Front-End Commands) allows users to process all 8 bits of information sent by the terminal for each character. Output onto BI files allows you to control the parity bit sent to your terminal. If you wish to construct 8-level paper tapes you will use BI files.

Instructions on the reading and writing of binary files are given in Appendix C.

5.1.5
BF (Binary Fancy) Files

BF stands for "binary fancy," which means that end-of-line and carriage-control conventions of ASCII-Fancy apply, but as in binary (BI), no translation is done.[1] The BF file is a connected file which sends alternate character set codes to a terminal using that character set (e.g., APL). When the terminal is switched to an alternate character set (such as APL) the Front-end will automatically translate all data written to DC or AF files. If you wish to send data in the alternate character set and use standard carriage controls, you must write to a BF file.

To write data in binary fancy, you must write to a file connected as a BF file (see Sections 5.2 and 5.8.5 for methods used to create a BF connected file).

Data should not be written to such a file unless the terminal is switched to the alternate character set (or the first line sent contains an SO character; see Section 8.2.4). Data on a BF file should be in the codes of the alternate character set, because no translation will be done.

Note: The following characters have no representation in the APL character set:

"	&
#	@
%	'

The double quotes will translate as a single quote: the remaining characters translate to nulls.

For discussion of alternate character sets, see Section 8.2.4.

5.2
CONNECT

You may connect files in one of the five modes described in Section 5.1 by means of the CONNECT command. This command connects the files specified. You may connect as many files as your file limit permits.

CONNECT,lfn$_1$[=mode][,...][,lfn$_n$[=mode].

lfn — the name of the local file to be connected.

[1] Binary Fancy files were created because APL needs them; users are not expected to need them. Just because you have an APL terminal doesn't mean you have to have BF files - you only need them if you want to send APL codes.

mode may be one of the following:

DC	Display Code; this is the default.
AS	ASCII
AF	ASCII Fancy
BI	Binary
BF	Binary Fancy
SAME	implies the connection of a previously-connected file in the mode last used.

Once a file is connected, all input/output operations that take place have no impact on the contents of the file. Read requests go directly to the program asking for input; write requests go directly to the user's terminal.

In fact, the file need not exist when the CONNECT occurs; nonexistent files are created, connected, and left empty until disconnected or returned.

The connecting of an already-connected file in DC, AS, AF, BI or BF mode causes it to be reconnected in the specified mode. Connecting such a file in SAME mode has no effect.

The CONNECT command allows the specification of multiple modes in a single command. For example,

CONNECT,A,B,C=AS,D=AF,E=BI,F=DC,G=SAME,H=BF.

would connect

A,B,F	DC
C	AS
D	AF
E	BI
G	SAME
H	BF

Of course, any one file can be connected in only one mode at a time.

5.3 SETCODE

The SETCODE command can be used to set the character code associated with a file either before or after it has been connected. (The character code set when a file is connected will override any previous code set by SETCODE.) The SETCODE function is specified as follows:

SETCODE,lfn_1=cc_1[,lfn_2=...].

lfn is the local file name.

cc is the character code: DC, AS, AF, BI or BF.

You can perform SETCODE from a FORTRAN program. The calling sequence is

CALL SETCODE (lfn,2Lcc)

lfn is the logical unit number (1—99) or the file name in L format.

cc is the character code: DC, AS, AF, BI or BF.

COMPASS programmers have access to the SETCODE macro, which can be called with

SETCODE lfn,cc[,r]

lfn is the address of a word containing the file name in L format. (Note that bit 0 is the complete bit and must be zero.)

cc is the character code: DC, AS, AF, BI or BF.

r recall parameter. If r is not blank, the request is made with recall.

5.4 DISCONT

DISCONT,lfn_1,...,lfn_n.

This command disconnects a previously connected file. The one form "DISCONT,lfn." is used to disconnect all connected file types. The files remain assigned to the terminal job after the DISCONT. If the files were created by the CONNECT command, they remain as empty local files.

A DISCONT of a nonexistent file causes the creation of that file. A DISCONT of a non-connected file has no effect.

5.5 PROMPT

The system does not normally indicate to the interactive user that a program is waiting for input. The PROMPT command asks the system to provide that information.

If you enter

PROMPT,ON. or PROMPT.

subsequent connected read operations will cause the system to send a carriage return, line feed, and asterisk as a prompt for input.

Prompting remains in effect for the rest of the terminal session unless disabled by

PROMPT,OFF.

PROMPT does not tell you what input the program desires—only that the program is awaiting response. System programs and user programs that expect interactive input usually display their own explanatory statements for prompting messages. If you anticipate the input required by a program and type it before the prompt is issued then no prompt will occur when the program reads the data.

Below is an example of the use of PROMPT.

```
OK-PROMPT.
OK-CONNECT INPUT.
OK-PNPURGE.
* MYPERMFILE
** EOP
OK-
```

Note that the system prints the asterisk prompt to indicate that PNPURGE is awaiting input from the connected file INPUT.

5.6 *EOS, *EOSnn, and *EOP

If a program is waiting for input, you may wish to enter end-of-section or end-of-partition. The following may be typed:

*EOS, *EOR } indicates end-of-section level 0.

*EOSnn, *EORnn } indicates end-of-section, level nn, where nn is an octal level number (0≤nn≤16).

*EOP, *EOF } indicates end-of-partition.

These special "commands" can contain no other characters and no trailing blanks.[1] If they are typed with other characters, they are merely passed to the program as normal data. See the example in Section 5.7.1.

*EOS17 will be interpreted as an end-of-partition.

Note that EDITOR can accept *EOS, *EOSnn and *EOP as values for lines. Such lines are converted to the appropriate record or file delimiter when a SAVE directive is executed.

5.7 Copy Utilities

You sometimes need a quick and simple means of adding information to a file interactively. One way is the use of EDITOR; another is the use of COPY. For example (user responses are shaded):

[1] In order to retain compatibility with previous systems, *EOR is equivalent to *EOS, *EORnn is equivalent to *EOSnn, and *EOF is equivalent to *EOP.

```
OK-[redacted]
OK-[redacted]
OK-[redacted]
*[redacted]
*[redacted]
*[redacted]
*[redacted]
OK-[redacted]
SUBMITTED UNDER SEQUENCE TB12345
OK-
```

Notice the use of two EOPs to terminate the COPY process. COPY will copy EOPs to a file until it encounters two EOPs in succession.

One limitation to this use of COPY is the fact that once a line is entered, it cannot be modified. Users can delete the last character typed with the backspace (CTRL-H), underscore or back arrow; similarly, users can erase a line in progress by entering CTRL-X. Once a line has been sent by pressing the carriage return, it cannot be changed. This process is therefore useful only for short copies where mistakes are unlikely.

COPY is the only file copying utility useful for this purpose. Other routines, such as COPYCR, COPYBR, COPYCF and COPYBF will not operate correctly on connected files and will abort.

5.8 Language Facilities

You can communicate interactively with programs written in any language available at MSU by connecting the appropriate files.

5.8.1 FORTRAN 4

FORTRAN Extended Version 4 allows the connection of files via the following subroutine calls:

```
CALL CONNEC(lfn)
CALL DISCON(lfn)
```

where the file, lfn, is indicated by: an integer constant, n, representing a logical unit number from 1 to 99; a Hollerith constant in the format nLfilename; or an integer variable containing any of the preceding forms. For example, to connect TAPE12, the programmer could use any of the following:

```
CALL CONNEC(12)

CALL CONNEC(6LTAPE12)

IN=12
CALL CONNEC(IN)

IN=6LTAPE12
CALL CONNEC(IN)
```

In order to connect a file as an ASCII file, the user specifies

```
CALL CONNEC(lfn,mode)
```

where mode is an integer variable or a constant of value 2LAS, 2LAF, 2LBI, 2LBF, 2LDC or 4LSAME.

For example:

```
      CALL CONNEC(6LTAPE12,2LAS)
```

or

```
      I=2LAS
      CALL CONNEC(6LTAPE12,I)
```

would connect TAPE12 as an ASCII file. If code is omitted, as in the earlier examples, the file is connected DC. If the code SAME is used, the file is connected in the same mode last used in this job.

You should note that the FTN call only allows one file to be connected or disconnected per call.

The FTN 4 input/output requests READ, WRITE, PRINT, and NAMELIST are useful only on DC connected files[1]. BUFFER IN statements made on connected files cause the transfer of one line of input from the terminal. You should take care to specify a buffer large enough to accommodate all the characters you will input. (The maximum number of characters that can be entered on one line is governed by the Front-End command INLEN. See Section 8.3.8.) Failure to do so will result in the loss of data for BUFFER IN statements; for other FTN input, a fatal-to-execution error will result.

FTN programs automatically connect the files INPUT and OUTPUT, unless the program establishes the labeled common block /CONECIO/. This is accomplished via

```
      COMMON/CONECIO/dummy
```

where dummy is a dummy variable. Note that execution-time diagnostics such as attempts to read past end-of-information will show up at the terminal only if OUTPUT is connected.

Following is a simple example program to be used interactively:

```
      PROGRAM FACT(INPUT,OUTPUT)
      COMMON/EXECMSG/TRASH
1000  CONTINUE
      PRINT*,' ENTER VALUE-'
      READ*,I
      IF(EOF(5LINPUT).NE.0)GOTO 9999
      N=1
      DO 1 J=1,I
      N=N*J
    1 CONTINUE
      PRINT*,N,' =FACTORIAL OF ',I
      GOTO 1000
9999  CALL EXITNM
      END
```

The program could easily be compiled and executed using EDITOR. Once loaded, the program would connect INPUT and OUTPUT. The list-directed input/output statements (READ,* and PRINT,*) are often used in programs to be run interactively. Note, however, that the above program could be executed from batch or interactive with the same results: from batch, the reads would come from the file INPUT, and the prints would appear on the file OUTPUT.

[1]The FASTIO routines on L*UNSUP provide a simpler mechanism for performance of ASCII input/output.

Assuming the object code for the program is on the local file, LGO, the user might use the program as follows:

```
OK-LGO
ENTER VALUE-1
 1=FACTORIAL OF 1
ENTER VALUE-6
 720=FACTORIAL OF 6
ENTER VALUE-*EOP.
OK-
```

Note the use of *EOP to signify the end-of-partition to the program.

FTN programs always check to see if a file to be written on is connected. If it is, the information is written with a WRITER request, ensuring the immediate transfer of the data to the terminal. FTN programs also always check to see if a file to be read from is connected. If it is, and illegal data is entered (say, a "Q" in an I4 field) the following message is output:

```
Q<ERROR, RETYPE RECORD AT THIS FIELD
```

This message will be reissued until a legal value is entered. Calls to ERRSET do not prevent the message. Once valid data is entered, the program continues as if valid data had been typed in the first place.

Note that FTN does not allow write operations on a file to be followed by read operations. For that reason, FTN programs reading from and writing on connected files must use separate files for the two operations, or you must be sure to rewind the file between the write and read operations.

In order to perform input/output on an ASCII or ASCII Fancy file, FTN programs must use BUFFER statements. For instance, in order to write an upper and lower case message at a terminal, you might write the following program:

```
      PROGRAM BUF(TAPE1,TAPE2,OUTPUT)
      DIMENSION MSG(3)
      DATA MSG/0040 0115 0151 0156 0164 B,
     + 0040 0112 0165 0154 0145 B,
     + 0160 0041 0041 0041 0000 B/

      CALL CONNEC (1)
      CALL CONNEC (2,2LAF)

      PRINT *,' READY FOR MSG?'
      READ(1,1)NRES
1     FORMAT(1R1)
      IF (NRES.NE.1RY)GOTO 99

      BUFFER OUT(2,0)(MSG(1),MSG(3))
      IF(UNIT(2))99,99,99

99    CONTINUE
      END
```

Note three different connected file operations here: First, the program automatically connects OUTPUT. The program then connects TAPE1 as a DC file and TAPE2 as an AF file. The PRINT,* appears on OUTPUT (and hence at the terminal). The READ is from TAPE1. If you type "Y" when prompted, the program will BUFFER OUT a single-space carriage control, a line feed, and send the characters "Mint Julep!!!" to the terminal.

Note that each set of data written via BUFFER OUT onto AF files begins with a carriage control: each BUFFER OUT is terminated at a system level with a WRITER request, which always implies end-of-line even if a record lacks the end-of-line character. (See Section 5.6.)

Although batch jobs may issue connect requests without error, users often want their programs to behave differently when run from batch than interactively. Programs may determine whether they are running in batch or interactive mode by using

```
CALL INTRCOM(J)
```

or

```
J=INTRCOM(x)
```

The variable J will be set to -1 (TRUE) if the program is running interactively; it will be set to 0 (FALSE) if the program is running from batch.

You may specify INTRCOM as a LOGICAL function as well:

```
LOGICAL INTRCOM
.
.
.
IF (INTRCOM(K))GOTO 20
```

In the above example, the branch would be taken only if the program were running interactively.

5.8.2 FORTRAN 5

The FTN 5 compiler may be invoked using the system command 'FTN5'.

Although EDITOR compilation directives are not officially available in FTN 5, a program on the Unsupported Library called XFTN5 provides directives which perform similar functions. Further documentation of XFTN5 may be obtained using the command:

```
HELP,L*UNSUP,XFTN5.
```

Consult the *CDC FORTRAN Version 5 Reference Manual* for documentation of FTN 5 conventions.

5.8.3 PASCAL

PASCAL programs do not automatically connect any files when run interactively. They do, however, issue WRITER requests whenever a line is to be output to a connected file, ensuring the immediate transfer of the data to the terminal, as is the case for FTN. However, there is a hindrance to the interactive use of PASCAL: the "next" character on any file to be read from is always defined. Hence, a RESET operation will cause a "pre-read" to be done on an input file, causing reads to precede the appropriate prompting message. To avoid this, the user might use the tactic illustrated in the following program:

```
PROGRAM KLUDGE(INP,OUTPUT);
(* THIS PROGRAM INTERACTIVELY SQUARES A GIVEN NUMBER *)
VAR
     I,J: INTEGER; INP: TEXT; FLAG: BOOLEAN;
BEGIN
(* INITIALIZE "FIRST READ" FLAG *)
FLAG := FALSE;
(* WE LOOP READING A NUMBER AND OUTPUTTING ITS SQUARE TILL EOF*)
REPEAT
     WRITELN(?INPUT?);
          IF NOT FLAG THEN BEGIN
               RESET(INP);
               IF NOT EOF(INP) THEN READ(INP,I);
               FLAG := TRUE;
               END
          ELSE BEGIN
               READLN(INP);
               IF NOT EOF(INP) THEN READ(INP,I);
          END(* IF *);
     IF NOT EOF(INP) THEN BEGIN
          J := I*I;
          WRITELN(I,?SQUARED=?,J);
     END(* IF *)
UNTIL (EOF(INP));
END.
```

In the program, the file INPUT is deliberately omitted; its presence would imply a pre-read, causing a read prior to the prompt message. By delaying the RESET until the prompt has been issued, this problem is avoided. The program will loop, prompting for input and listing the square of each integer entered:

```
OK-CONNECT,INP,OUTPUT,
OK-LGO.
INPUT-3
3 SQUARED= 9
INPUT-7
7 SQUARED= 49
INPUT-*EOF
OK-
```

Note the connecting of INP and OUTPUT prior to the load request so that input will be initiated at the terminal and output will be sent to the terminal.

PASCAL programs can easily accommodate ASCII I/O by use of a type declaration such as:

```
TYPSYSASC=0..4095;
TYPASC=0..127;
TYPASCBYT=1..5;
TYPASCWRD=PACKED ARRAY [TYPASCBYT] OF TYPSYSASC;
```

An ASCII file would then be declared in the variable declaration part in this manner:

```
INFILE: SEGMENTED FILE OF TYPASCWRD;
```

Reads (or writes) would be done through GETs or PUTs:

```
   PROCEDURE GETCHAR;
(* ASSUME NXTIN IS A GLOBAL VARIABLE THAT
   HAS BEEN INITIALIZED TO 1 *)
   BEGIN
   IF NXTIN=5 THEN
        BEGIN
        GET(INFILE);
        NXTIN:=1;
        END(*BEGIN*)
   ELSE
        BEGIN
        NXTIN:=NXTIN+1;
        END;
(* ASSUME -CURCHAR- IS THE CHARACTER TO BE
   GOTTEN FROM THE FILE *)
   CURCHAR:=INFILE↑[NXTIN];
   END(*GETCHAR*);
```

A similar strategy could be used for ASCII output in PASCAL.

5.8.4 BASIC

BASIC users generally will use PRINT and INPUT statements for connected I/O. For example,

```
100   REM THIS PROGRAM COMPUTES FIBONACCI NUMBERS .
110   REM IT COMPUTES THE FIRST -N- NUMBERS, WHERE N
120   REM IS A LIMIT THE USER SPECIFIES.
130   PRINT "A NEGATIVE NUMBER TERMINATES INPUT..."
140   PRINT "ENTER";
150   INPUT L
160   IF L < 0 THEN 300
170   PRINT "FIRST", L, "FIBONACCI NUMBERS"
180   LET F=0
190   PRINT F
200   IF L=1 THEN 140
210   LET S=1
220   PRINT S
225   IF L=2 THEN 140
230   FOR I=3 TO L STEP 1
240   LET N=F+S
250   PRINT N
260   LET F=S
270   LET S=N
280   NEXT I
290   GOTO 140
300   END
```

The program will output the prompt "ENTER" followed by a question mark, which is inserted by BASIC. The semicolon at the end of the PRINT causes the question mark prompt to appear on the same line as the "ENTER". If you chose to omit the printing of a descriptive prompt, BASIC would output the question mark on a line by itself. The program as listed produces results as follows:

```
OK-BASIC
A NEGATIVE NUMBER TERMINATES INPUT...
ENTER?1
FIRST     1          FIBONACCI NUMBERS
0
ENTER?6
FIRST     6          FIBONACCI NUMBERS
0
1
1
2
3
5
ENTER?-10
OK-
```

If in response to a connected read you enter too much or too little data, BASIC displays an appropriate message and reissues a prompt. Avoid using *EOP (or *EOF); BASIC will always interpret the entering of *EOP (or *EOF) as too little data.

BASIC provides no statement for file connection and disconnection. However, BASIC does automatically connect OUTPUT for PRINT and INPUT for INPUT operations.

5.8.5 COBOL

COBOL users perform DC connected I/O via ACCEPT and DISPLAY verbs. These I/O statements automatically connect a local file named TERMINAL. (READ and WRITE statements may be used, but you must connect files as appropriate.)

The relevant parts of an interactive COBOL program might look like this:

```
.
.
.
ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SPECIAL-NAMES.

   TERMINAL IS TTY.

.
.
.
PROCEDURE DIVISION.
.
.
.
   ACCEPT NUM FROM TTY.
   MULTIPLY NUM BY NUM GIVING SQUARE.
   DISPLAY "THE SQUARE IS" SQUARE UPON TTY.
```

The ACCEPT statement automatically generates the prompt

ENTER COBOL INPUT

ASCII I/O in COBOL is most practically accomplished through the use of an FTN subroutine that sets up files via Cyber Record Manager and then performs the necessary BUFFER operations.

Like BASIC, COBOL provides no statements for file connection/disconnection.

5.8.6 COMPASS

COMPASS programmers connect files by using

CONNECT stwrd, R, code

stwrd is the address containing the file name in L format.

R specifies recall.

code is:
AS for ASCII
DC for Display Code
AF for ASCII Fancy
BI for Binary
BF for Binary Fancy
SAME for the mode last used in this job for this file.

Files are disconnected via

DISCONT stwrd, R

The COMPASS programmer must be aware of two distinct system read operations: READ and READSKP. Either will read one line per request from the terminal. The distinction lies in the fact that if more characters are entered than will fit in the buffer, a READ will result in error code 10B, and a READSKP will not result in any error. (Error code 10B is "device capacity exceeded.".)

Programs can output as many lines as will fit in the program buffer on any WRITE operation. COMPASS programmers should note that a WRITER or WRITEF operation on a connected DC or AF file always implies an end-of-line, whereas a WRITE does not. Thus the first character output on a given WRITE will be used for a carriage control if the last operation was a WRITER and/or if the last character output was an end-of-line. Otherwise, the first character will be sent to the terminal. Be careful to end each set of data output via WRITE with an end-of-line unless you intend to write a partial record.

When performing interactive I/O from COMPASS, specify sufficient room for the longest possible record in the FET and buffer. If up to N characters may be processed, N/10+2 words are needed for DC files; N/5+2 words should be allotted for AF, AS or BI files. DC and AF output lines must have the appropirate carriage control; AS output lines must have the desired carriage return/line feeds/delays combination. Whether the file is connected DC or AS, COMPASS programmers will generally use a WRITE or WRITER request so that output to a connected file will show immediately. Note that DC and AF output lines must be terminated with 12 bits of binary zeros.

In general, COMPASS programmers will specify recall for connected I/O to ensure the appropriate order of operations. Completion of a write operation to a connected file does not imply that data has, in fact, finished printing at the user's terminal; rather, completion of a write operation merely implies that the system has read all of the information from the user program's output buffer.

In order to see if a file is connected, as well as its last status of connection, the COMPASS programmer uses FNTSTAT:

```
FNTSTAT          blok, R
```

This macro call returns various items of information in the block blok. The user can check the device type in bits 59-48 of word 2 of blok. If the device type is 61 octal, the file is connected. How the file is connected is revealed by the disposition code found in bits 35-24 of word 2.

The field consisting of bits 30 through 32 of word 2 is set to 0 if the file is a DC-connected file; it is set to 2 if the file is connected AS; 3 for AF, 4 for BI, and 5 for BF. Note that the disposition code remains set even if the file is disconnected. Users can thus determine the last mode of connection of a given file in a given job.

The block used by FNTSTAT may be established via

```
blkname     FNTBLOK          fname,length
```

blkname is the address of the block.

fname is the file for which information is sought.

length is the length of block to be generated.

FNTSTAT and FNTBLOK are documented in the *SCOPE/HUSTLER Reference Manual*, Section 8.5.16.

COMPASS users can call INTRCOM to determine whether a program is running interactively or from batch with a simple macro call:

```
INTRCOM          parm, R
```

where parm is a location that has been set to zero. The system will set parm to the value of 3 if the job is being run interactively; to 1 if not. Bits 24-35 will be set to the interactive line length. Bits 36-39 will be set to 1 if interactive; 0 if batch.

As a COMPASS programmer you may wish to have your program take action if a user of that program fails to enter any data after a specified number of seconds elapse. The TIMEOUT function allows a program to regain control in such cases. TIMEOUT is called by:

```
TIMEOUT          status
```

Status is set prior to this statement by the following call:

```
VFD              36/0,12/count,12/0
```

where count represents the number of seconds the system should wait before it swaps the program back in.

TIMEOUT should always be followed by a read request. The read will cause the program to be swapped until the time limit count is reached, or the user has entered data. (Timing begins as soon as the last output line begins to print.)

If after the specified number of seconds no data has been entered, the system will swap the program in and place a special code in the input buffer.

74_8 is returned for DC files; 2015_8 for AS, AF or BI files. Since a 74_8 can be entered using the @ sign, 74_8 is used for upward compatibility with previous systems. There is currently no way to uniquely determine that the time has elapsed with DC files.

```
      PROGRAM FLUSTER(INPUT,OUTPUT)
      CALL NOBLANK
      PRINT*,"YOU HAVE 15 SECONDS TO TELL ME YOUR AGE"
* CALL COMPASS ROUTINE TO MAKE TIMEOUT CALL.
      CALL TIME
      READ 10,INP
10    FORMAT(A10)
*  IF FIRST CHAR IS 74 OCTAL, TIME LIMIT WAS REACHED
* BRANCH IF SO.
      IF(SHIFT((INP.AND.MASK(6)),6).EQ.74B)GOTO 2
* WE WERE GIVEN AN AGE.  WE TRANSLATE TO
* INTEGER FROM DISPLAY CODE:
      DECODE(5,20,INP)INP
20    FORMAT(I5)
* IF AGE IS GREATER THAN THREE, WE'LL CALL THE USER OLD
      IF(INP.LE.3)GOTO 1
* IT'S AN OLDIE
      PRINT *,"WOW.  AN OLDIE"
      STOP
1     PRINT *,"THAT-S NOT SO OLD"
      STOP
* NOTHING WAS ENTERED.  WE COMPLAIN.
2     PRINT *," TOO OLD TO TYPE?"
      END
          IDENT    TIME              PLANT A TIMEOUT REQUEST
          ENTRY    TIME
STAT      VFD      36/0,12/15,12/0 STATUS WORD W/ 15 SEC. LIMIT
TIME      BSSZ     1                 ENTRY POINT
* PLANT TIMEOUT REQUEST. NOTE THAT FORTRAN READ WILL
* FOLLOW WHEN WE RETURN TO MAIN PROGRAM
          TIMEOUT  STAT              MAKE REQUEST...
          EQ       TIME              ...AND RETURN TO MAIN PROGRAM
          END
```

6

Debugging Aids

Debugging programs interactively is often faster and more convenient than debugging programs using batch processing. An obvious advantage is the interactive user's ability to correct and re-test programs immediately. This chapter will not delve into the differences between batch and interactive systems, but will describe control statements used to obtain information at a terminal that will be helpful in determining the cause of program errors.

The analytical aids described in this chapter are divided into the following categories:

Compilation aids

Debugging aids are available for the programmer but their use is restricted to certain compilers. ERRS, a routine which prints an error summary, fits into this category. To check on applicable uses with compilers, see Section 6.1.1.

Reference maps are compiler-generated lists of program elements. 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.

System Error Messages

Control statements executed in an interactive session are listed along with system diagnostics and accounting messages in a file called the dayfile. The DAYFILE statement saves dayfile messages on a local file and displays selected line ranges from this file at an interactive terminal. This is explained in more detail in Section 6.2.1

Execution-time Error Detection

You can interactively debug programs while they are executing using Cyber Interactive Debug. Errors encountered during execution may also be detected by dumps, which display of the current contents of the central processor registers and selected memory areas. The statements DMP and SAVEDMP generate information of this type. Section 6.3 describes facilities and commands used to perform these tasks.

Loader Error Detection

Errors may be detected using load maps. A load map is a listing showing the location of each subprogram and common block within the central memory field length assigned to the job. Referenced subroutines not available for loading (unsatisfied externals) are also listed. MAP and TRAP are loader-related control statements and affect loader performance. The MAP statement generates a loader map for each program subsequently loaded. TRAP is a set of programs which executes simultaneously with the user program. For more information see Section 6.4.

Job Processing Alternatives

The EXIT and MODE statements control the execution of an interactive job after an error has been detected. For more details see Section 6.5.

6.1 Compilation Aids

Certain compilers allow the use of control statements to aid in finding and remedying compiler errors.

6.1.1 ERRS

Programmers can use the ERRS statement to scan a program for compiler errors. Currently, ERRS can process listings from FTN 4 (except FTN,TS), SPSS, COBOL, COMPASS, and UPDATE. When a program has a line containing 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 listing of the fatal error messages and the sequence number(s) of the line(s) to which each message applies.

The ERRS statement is implicitly executed whenever an EDITOR compiler directive is given (see Section 3.10 and example below). The following compiler directives do not execute ERRS: BASIC, BASICX, COBOLX, COMPX, and FTNX. ERRS can also be explicitly executed. The control statement is shown below, followed by descriptions of optional parameters, which are grouped according to their function.

ERRS[,optional parameters].

Input/Output Specifications

I=inlfn specifies the input file, which ERRS will search for program listings. This file will be disconnected and rewound before and after ERRS processing. The default input file is OUTPUT.

O=outlfn specifies the output file, on which ERRS will write the error summaries. The interactive default is TTYTTY, a special connected file used primarily as an interface between system programs and the user terminal. **Caution:** If the input and output files are the same, the program listings, as well as any other information on the input file, will be destroyed.

Listing Options

ALL instructs ERRS to include non-fatal diagnostic messages in the error summaries. The default displays fatal error diagnostics only.

F instructs ERRS to print a full error summary, as opposed to a short error summary as specified by the S parameter. The error summary is a listing of all lines in which errors occurred.

NI eliminates the informative error messages if no fatal error diagnostics are found. The NI parameter is ignored if the ALL parameter is specified.

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 in their entirety). This is the default. (Exception: 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.)

Processing Options

NA requests that ERRS not abort if errors are found in the listing.

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).

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.

Caution: If ERRS finds any fatal errors, it will end with an abort request after completing the error summaries (unless the NA parameter is specified). If you want to continue job execution in such cases, the NA parameter should be used.

Example:

The FTN 4 listing of a sample program which contains several errors is shown below. Following it is output from ERRS illustrating the effects of various control statement options.

```
100= PROGRAM EXAMPLE (INPUT,OUTPUT)
110=C THIS PROGRAM IS SUPPOSED TO READ IN 10 NUMBERS AND
120=C PRINT OUT THE TOTAL AND AVERAGE OF THE NUMBERS.
130= DEMENSION A(10)
140= I=1
150= TOTAL=0
160=10 READ 100,A(I)
170=100 FORMAT (F10.2)
180= TOTAL=TOTAL+A(I)
190= I=+1 190
200= IF I GT 10 ) GO TO 20
210= GO TO 10
220= AVERAGE=TOTAL/10
230=20 PRINT 200,TOTAL,AVERAGE
240=200 FORMAT (* *,2F10.2)
250= END
OK-ftn.
 COMPILING EXAMPLE
          3  FORTRAN ERRORS IN  EXAMPLE
        .051 CP SECONDS COMPILATION TIME
CPU ABORT
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
OK-errs,f,all,ns.
CPU ABORT
1 INFO ERRORS IN EXAMPLE
ERRORS WERE FOUND IN THE FOLLOWING LINES:
4= DEMENSION 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
```

The EDITOR compilation directive 'FTN' executes a pre-defined sequence of directives and commands. 'FTN.' and 'ERRS,S.' have identical output because 'FTN.' initiates ERRS automatically after compiling by executing the following commands.

```
SAVE,SETFILE[,lnum][,text].
REWIND,LGO.
FTN,I=SETFILE,T.
ERRS.
```

As shown, an additional call for 'ERRS,S.' after initiating 'FTN.' would be redundant.

```
OK-errs,s.
CPU ABORT
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
OK-
```

In the second sample note that a line count is used instead of the sequence numbers embedded in the lines.

Note: FORTRAN 5 syntax may vary from FORTRAN 4. Refer to the *CDC FORTRAN Version 5 Reference Manual* for further discussion.

6.2 System Error Messages

Messages from the computer to the user are often helpful in locating the source of errors. The dayfile provides a listing for basically that reason.

6.2.1 DAYFILE

When a job begins execution, the system creates a file to collect dayfile messages. As each control statement is executed, the statement image is copied to this "dayfile", allowing the user to trace the progress of the job. Between these dayfile messages are recorded various system statistics, error diagnostics, and informative messages. The final lines of the dayfile give a partial summary of the job cost.

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 batch OUTPUT file.

The DAYFILE statement has two formats.

Format 1: DAYFILE[,F].
Format 2: DAYFILE,fromline,toline[,F].

Format 1 saves on file DAYF all dayfile messages accumulated since the previous format 1 DAYFILE command or the start of the interactive session. 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. It must be preceded by a DAYFILE command in format 1. Format 2 reads only the current copy of DAYF; it does not update this file with new messages.

F — full option; all of the lines are displayed. If omitted, the CP, PP, NL, RP and other accounting messages issued 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.

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 with 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 your dayfile and 16 words of somebody else's dayfile (garbage). The DAYFILE routine sorts out this mixture and prints only the lines belonging to the user.

3. Because the accounting summary is not generated until job termination, these messages cannot be copied to DAYF.

4. DAYFILE connects file OUTPUT, allowing the dayfile messages to be displayed at the terminal.

Examples:

1. Below are samples of the DAYFILE command. (What you type-in is shaded.)

```
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 4, 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.NL        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, MTLY017PART2
             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
```

2. When a batch job is submitted from an interactive terminal, you may wish to catalog the job dayfile (and possibly the job output), so that the dayfile can be examined to determine whether the job executed properly before printing. Here is an example:

```
OK-SYSTEM,BATCH.
OK-N.
100=id,PNpn.
.
.
.
180=EXIT,S,C.
190=DAYFILE.
200=CATALOG,DAYF,JALDAYF,RP=2.
210=DAYFILE,1,999,F.
220=*EOR
.
.
.
500=*EOF
501==
EON-PROCESSING TEXT
OK-SAVE,A.
OK-DISPOSE,A,IN.
OK-
```

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, you could attach JALDAYF as local file DAYF and execute format 2 of DAYFILE to examine it.

```
OK-RETURN,DAYF.
OK-ATTACH,DAYF,JALDAYF.
OK-DAYFILE,6,999.
.
.
.
OK-PURGE,DAYF.
OK-
```

6.3
Execution-Time Error Detection

This section introduces Cyber Interactive Debug, a facility which allows you to interactively debug programs while they are executing.

Programming errors encountered during execution may be diagnosed by intermediate printouts, interactive debug techniques and dumps. The statements DMP and SAVEDMP generate selected dump information.

A dump is a printed listing of the contents of the arithmetic registers and a predetermined number of words within central memory that are assigned to a job. The contents of the registers are helpful in detecting false values used in calculations. The contents of the individual central memory words are meaningless without the use of a program map. When requesting memory dumps you must be certain that the field length (central memory used by your job) will not be relinquished to another user before the dump request is processed.

6.3.1 Requesting Memory Dumps

When requesting memory dumps the user must be certain that the field length will not be relinquished to another user before the dump request is processed.

For this reason, DMP and EXIT commands should be issued on the same line as the Load and Execute commands, as below:

```
FTN,I=COMPILE,L=LIST.LGO.EXIT.DMP.DMP,0,45000.
```

Another method of achieving the same thing would be to insert these statements (one per line) into a control statement file constructed under EDITOR and executed by the EXEC command (see Section 9.1) or GO directive (see Section 3.10.6). An alternative to DMP, the SAVEDMP command, instructs the interactive system to write a full binary dump to the file, TTYDMP, automatically upon abnormal termination of a user program.

6.3.2 DMP

The DMP statement generates a printed copy of all or part of the job's field allocation of central memory (known as the job's field length). Each word of memory is displayed as twenty 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 file OUTPUT, which is first disconnected.

The DMP statement has three forms, which can be represented as:

```
DMP.
DMP,fromline,toline.
DMP,toline.
```

fromline — the starting address, relative to the job field length. The first word of the field length is address 0.

toline — the last word address of the dump, relative to the job field length. The form 'DMP,to.' is equivalent to 'DMP,0,toline.' The dump is always restricted to your field length. DMP automatically corrects the dump limits if the specified limits are out of bounds.

The form 'DMP.' or 'DMP,0,0.' 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.

Note: DMP alters words (70-77) and one word per DMP parameter starting at word 2 in the dump.

```
DUMP    EXCHANGE PACKAGE          DMP,0,0.

 P   000251   A0  000500   B0  000000
 RA  164300   A1  000066   B1  000001  C(A1)=  4000  0000  0000  0000  0111    C(B1)=  0516  0420  0000  0000  0000
 FL  000500   A2  000120   B2  000000  C(A2)=  0516  0460  0000  0000  0000    C(B2)=  0000  0000  0000  0000  0000
 EM  070000   A3  000057   B3  000114  C(A3)=  0000  0000  0000  0000  0000    C(B3)=  5110  0001  1110  7114  6000
 RE  000000   A4  000001   B4  000000  C(A4)=  0516  0420  0000  0000  0000    C(B4)=  0000  0000  0000  0000  0000
 FE  000000   A5  000114   B5  000114  C(A5)=  5110  0001  1110  7114  6000    C(B5)=  5110  0001  1110  7114  6000
 MA  001200   A6  000001   B6  000500  C(A6)=  0516  0420  0000  0000  0000    C(B6)=    * * OUT OF RANGE * *
              A7  000372   B7  027761  C(A7)=  1722  6220  7717  4015  6063    C(B7)=    * * OUT OF RANGE * *

 X0  7777  7777  7777  7700  0000
 X1  4000  0000  0000  0000  0111
 X2  0516  0420  0000  0000  0000
 X3  0000  0000  0000  0000  0000
 X4  0000  0000  0000  0000  0000
 X5  6000  0000  0000  0000  0000
 X6  0400  0001  2100  0000  0000
 X7  1722  6220  7717  4015  6063

       00000   00000 00000 00000 00000          05160 42000 00000 00000          33000 00000 00000 00001          33000 00000 00000 00
017
       00004   00000 00000 00000 00000    00054=56110 03110 00054 54710          51100 00001 03110 00055          64550 02550 00000 46
000
       00057   00000 00000 00000 00000
       00060   15051 52000 00000 00061          00000 00500 00000 00001          51600 00001 04000 00054          63310 02300 00000 46
000
       00064   00000 00000 00000 00002          00000 00000 00000 00403          40000 00000 00000 00111          00000 00000 00000 00
000
       00070   04152 05633 56335 70000          00000 00000 00000 00000    00100=54000 00000 01000 00001

       00151   20152 63410 54111 53210          20152 53010 20152 63710          54111 53510 20152 53410
       00154   20152 53310 54111 10211          54111 10311 54111 10411          54111 10511 63160 46000          04000 00143 61000 46
000
       00160   00000 00000 00000 00371          00000 00000 00000 00000          00000 00100 05000 11762          00000 00000 01140 00
500
       00164   00027 76100 !
```

Figure 6.1: Standard Dump

Figure 6.1 shows a sample dump, produced by the statement 'DMP.'

6.3.3 SAVEDMP

The SAVEDMP command, an alternative to DMP, causes the system to write a full dump on the file TTYDMP, whenever there is an abno.mal termination of a user program.

The format of the SAVEDMP statement is:

SAVEDMP [,ON|,OFF].

ON Turns SAVEDMP on until a dump of the user's program is produced. If 'SAVEDMP.' is specified 'SAVEDMP,ON.' is assumed.

OFF Suppresses the automatic function. This is the default condition.

TTYDMP is a binary file which contains the exchange package contents and a copy of all memory locations within the program's field length. Before each load, SAVEDMP must be re-entered, but the file TTYDMP need not be returned or rewound.[1]

Example:

Consider the following command lines:

1. FTN.SAVEDMP.LGO.EXIT.DMP,20000.
2. FTN.SAVEDMP.LGO.

Line 1 generates an ordinary octal dump on OUTPUT as well as a binary dump on TTYDMP if the program terminates abnormally. Line 2 does not produce a dump on OUTPUT, but writes a binary dump on TTYDMP if the program terminates abnormally.

6.3.4 Cyber Interactive Debug (CID)

The Cyber Interactive Debug facility (CID) is an extremely powerful tool that you can use interactively to debug programs while they are executing. Standard relocatable programs and overlayed programs can be debugged using CID; segmented programs cannot.

CID allows you to use source variable names, source line numbers, and source-language-like statements when debugging programs. Breakpoints and traps can be set and cleared; memory and register contents can be examined and modified; execution-time errors, such as mode errors, are automatically trapped; execution can be suspended to allow files to be attached, for instance, then resumed at the same point sometime later in the session (or in another session, if all of CID's files are saved).

---

[1] TTYFDMP is a general purpose file display utility intended for interactive display and modificaton of a file which may be used with the file generated by the 'SAVEDMP' command.

TTYFDMP resides on the Unsupported library and is not officially supported by the Computer Laboratory. For further information on this utility, use the 'HELP' command in the form:

HELP,L*UNSUP,TTYFDMP.

CID is most effective when used in conjunction with object code specially designed for use with the debug utilities. Such code is generated automatically by the FTN 4, FTN 5 and BASIC compilers when DEBUG mode is in effect. This code can also be generated by these compilers by selecting various compiler call options (see the individual compiler reference manuals for details).

Cyber Interactive Debug is activated using the DEBUG control statement. The format of the DEBUG statement is:

DEBUG[,ON|,OFF|RESUME[,lfn]].

ON Default; activates debug mode. Whenever a relocatable binary program is loaded and executed, CID is loaded and given control.

OFF Terminates debug mode

RESUME Resumes the debug session suspended by returning control to the command mode of the operating system (see the SUSPEND command).

When the job is in the debug mode, the loader will load various CID control modules with every relocatable load. The loader will also create files containing load map and symbol table information for CID's use. CID obtains control from the loader and issues the message "CYBER INTERACTIVE DEBUG," then prompts with "?".

You may then enter CID commands that set breakpoints and traps, specify output options or preset any data values. A command 'GO.' starts execution of the object program. When any of the specified conditions occur, the condition is reported, execution is suspended and control passes to you at the terminal. Diagnostics, trap and breakpoint reports are displayed. During this time, you can use CID facilities to explore the behavior of your program. Program execution resumes at the location you specify. Any abnormal program abort, as well as normal program termination, returns control to CID. The command 'QUIT.' is used to terminate CID control.

Consult the CDC *Cyber Interactive Debug Reference Manual* for detailed information on the CID facility.

6.4 Loader Error Detection

Programming errors are sometimes encountered during loading sequences. The control statements listed below are loader-related and affect loader performance.

6.4.1 MAP

The MAP command is used to generate a loader map for each program subsequently loaded. A loader map is a listing of subroutine and common block locations within the central memory field length assigned to the job. Maps are normally written on local file MAPFILE, which is not rewound between loads. Unsatisfied externals, referenced externals not available for loading, are listed at the terminal.

The format of the MAP command is:

MAP,mapflag.

where mapflag may be specified by any of the following:

OFF the default condition. No loader maps are created.

```
LOAD MAP - TEST                          CYBER LOADER 1.1-428      02/19/79 .15.46.29.      PAGE      1


FWA OF THE LOAD              111
LWA+1 OF THE LOAD            403

TRANSFER ADDRESS -- TEST                 114


PROGRAM AND BLOCK ASSIGNMENTS.

BLOCK       ADDRESS    LENGTH      FILE      DATE      PROCSSR VER LEVEL  HARDWARE   COMMENTS

/A/             111         2
/B/             113         1
TEST            114         5      LGO
SUB             121         3      LGO
CPC             124       245   SL-NUCLEUS
 //             371        12


ENTRY POINTS.

ENTRY          ADDRESS          PROGRAM    REFERENCES

SUB                121          SUB        TEST              115
CPC                172          CPC        TEST              117


      .066 CP SECONDS              12200B CM STORAGE USED                    1 TABLE MOVE
```

Figure 6.2: Loader Map

ON writes a loader map including: loader statistics, block names and addresses, external/entry point cross-references.

PART writes a loader map that excludes entry point addresses.

FULL writes a loader map which includes the same information as 'MAP,ON'. plus a list of all entry point names and addresses including those which are not referenced.

MAPFILE may be listed at the terminal with LISTTY, or disposed to a line printer.

Figure 6.2 shows an example of a loader map produced by 'MAP,ON.' or 'MAP,FULL.'

For a further description and information see the CDC *Cyber Loader Reference Manual.*

6.4.2 TRAP

TRAP is a set of programs which executes at the same time as your program. This allows TRAP to process dumps while the program is executing. This is particularly useful in finding errors which are not immediately fatal and which may become obscured by the time the program actually stops.

Much of the information needed by TRAP is available only during the process of loading the program to be "trapped," so TRAP must work with a loader program. The loader is a system program that places the object code into central memory and makes it ready for execution. The TRAP routine uses information left on a local file by the loader to implement its directives.

The TRAP statement should immediately precede the load sequence to which it applies. Any number of TRAP statements are allowed. The TRAP routine reads and interprets directives from the input file and causes TRAPPER, the TRAP execution time routine, to be loaded with your program. TRAPPER is the first routine loaded.

The output is written on a listing 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.

In a TRAP routine, you can request a FRAME, or snapshot dump, of selected areas of memory whenever an instruction in a specified location is executed. The code can also be tracked, using TRACK. TRACK dumps registers and memory locations which are altered by the execution of instructions within a specified range of locations.

FRAME Output:

FRAME output consists of a dump of all the registers (if requested), and a core dump of the area specified in the directives. Only the contents of the P register appear in the dump if all the registers are not specified. 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 instructions in the range, 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 *Cyber Loader Reference Manual* (CDC publication 60344200).

6.5 Job Processing Alternatives

The control statements listed here are alternatives to the dumps, maps and listings stated earlier. Not fitting under any one of those categories, they still provide means for evaluating programming errors.

6.5.1 EXIT

The EXIT statement establishes EXIT routines that will be executed in case of an abnormal termination. When a fatal error is encountered, the system begins processing those commands which follow the next EXIT command in the command string or control statement file. Normal termination of the command sequence is the default condition. Several EXIT commands may appear within a sequence to establish a series of EXIT routines, as shown below:

```
FTN,I=A.LGO.EXIT,S.REWIND,LGO.FTN,I=B.LGO.EXIT,S.REWIND,LGO.FTN,I=C.LGO.
```

If the program on file A aborts, the system will compile and execute file B; and if that, too, aborts, the system will compile and execute file C.

The format of the EXIT control statement is:

```
EXIT[,option].
```

If 'EXIT.' alone is given in a command string, the following actions will take place: if no error occurs, execution of the command string will terminate when 'EXIT.' is encountered.

The options described below cause varied results depending on job error status at the time the above statement is encountered.

S allows processing to continue even after a control statement error occurs; the command string 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 command string. If an error occurs, 'C' has no effect.

U If 'EXIT,U.' is executed normally, the job will terminate. If an error occurs, 'U' has no effect.

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 in the command string; "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 Processing	Resume Processing	End Processing	End Processing	Resume Processing
Special errors (see list below)	Skip	Skip	Resume Processing	Skip	Resume Processing
Execution error (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 exceed — job has used all central processor time, money (job cost), files, or mass storage that it was allotted.
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 — 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.

Additional information on the EXIT command may be found in Chapter 7 of the *SCOPE/HUSTLER Reference Manual*.

6.5.2 MODE

The MODE statement allows a job to continue processing after encountering specified mode errors. A mode error may be any:

1. Reference to an address outside the field length of the job; such an address may be generated during execution if a nonexistent subroutine is referenced.

2. Reference to a floating point operand (any number used in a calculation) that has an infinite value.

3. Reference to a floating point operand for floating point arithmetic which has an indefinite value.

Normally, these errors will terminate processing; 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 the control statements are executed.

The MODE control statement has the following format:

MODE,n.

n is a number specifying the halt conditions:

0 No halt occurs; all mode errors are ignored.

1 Halt only if address is out of range (mode 1).

2 Halt only if operand is a floating point number of infinite value (mode 2).

3 Halt if address is out of range or operand is infinite (mode 1 or 2).

4 Halt only if operand is floating point number of indefinite value (mode 4).

5 Halt if address is out of range or operand is floating point number of indefinite value (mode 1 or 4).

6 Halt if operand is a floating point number of infinite value or a floating point number of indefinite value (mode 2 or 4).

7 Halt if operand is a floating point number of infinite value or floating point number of indefinite value or address is out of range (mode 1, 2, or 4). This is the default. Termination occurs whenever any of the three basic errors is detected.

Once a MODE statement is encountered, it will remain in effect until the interactive session 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 store information outside your allotted memory appears to complete normally; however, no storage occurs. When an attempt is made to access information outside your allotted memory, a value of zero is returned to the program.

Below is an example of a program containing a mode error:

```
100= PROGRAM MODERR (OUTPUT)
110= X=3.14159
120= Y=0.0
130= Z=X/Y
140= PRINT 1,X,Y,Z
150= ERROR=Z+1
160= PRINT 1,ERROR
170=1 FORMAT(3F10.5)
180= END
OK-ftn.lgo.
 COMPILING MODERR
       .095 CP SECONDS COMPILATION TIME
 EXEC BEGUN.10.22.49.
  3.14159    0.00000          R
*ERROR MODE 2 AT 002203 (INFINITE FLOATING POINT OPERAND)
OK-mode,0.
OK-ftn.lgo.
 COMPILING MODERR
       .113 CP SECONDS COMPILATION TIME
 EXEC BEGUN.10.23.42.
  3.14159    0.00000          R
        R
      END  MODERR
       .005 CP SECONDS EXECUTION TIME
OK-list,f.
100=       PROGRAM MODERR (OUTPUT)
110=       X=3.14159
120=       Y=0.0
130=       Z=X/Y
140=       PRINT 1,X,Y,Z
150=       ERROR=Z+1
160=       PRINT 1,ERROR
170=1      FORMAT(3F10.5)
180=       END
OK-mode,2.
OK-ftn.lgo.
 COMPILING MODERR
       .105 CP SECONDS COMPILATION TIME
 EXEC BEGUN.10.32.17.
  3.14159    0.00000          R
*ERROR MODE 2 AT 002203 (INFINITE FLOATING POINT OPERAND)
OK-
```

7

DISPOSE — Routing Files for Off-Line Processing

The SCOPE/HUSTLER system allows you to send files from your terminal to the central site for batch processing, or to off-line output devices for printing or card punching. Both operations are accomplished by the DISPOSE utility. The first section of this chapter deals with the syntax of the DISPOSE command and how it is used to route output. The second section deals with submitting batch jobs from an interactive terminal.

7.1 Routing Files for Off-line Processing

The DISPOSE command directs a temporary file to a destination, which may specify the input queue, the card punch, or a particular line printer. The file need not be rewound before it is disposed. After you enter a DISPOSE command, the system responds with the sequence number under which the output is to be picked up. For example,

```
SUBMITTED UNDER SEQUENCE SB66974
```

The interactive form of the DISPOSE command is:

```
DISPOSE,lfn,dis[=dest][,C=ncopies][,L=lmt][,I=acct].
```

lfn — the local file name. This cannot be a permanent file or the special file, INPUT.

When DISPOSE is processed the file is immediately released from the job (unless the lfn is preceded by an asterisk). Any further reference to this file name will either create a new file or cause an error.

dis — disposition: Same as for batch.

- PAF print the file on 96-character ASCII Fancy upper/lower case printer only. The file may contain either Display code or ASCII characters or both.
- PAU print the file on 64-character, ASCII Fancy upper case printer only. Any lower case characters in the file will be mapped to upper case equivalents. The file may contain either Display code or ASCII characters or both.
- PA print the file on any ASCII printer. The file may contain either Display code or ASCII characters or both.
- PR print the file on any available printer at the specified destination. Note: If an ASCII Fancy file is sent to a line printer intended to print Display Code only, the output will be unreadable.
- PU punch the file in Hollerith mode (026).
- PB punch the file in binary mode.
- P8 punch the file in 80 column binary mode.

P9 punch the file using 029 keypunch codes.
PC punch only the first 80 columns of each record in the file.
IN place the file in the input queue (see Section 7.2), and execute it as a standard batch job. Note that the first line of this file must be a legal job card with the additional PNxxxxxxx parameter, or the appropriate job card for Wayne State or University of Michigan if the job is disposed through the Merit Network.

dest an optional destination parameter for print files. If not specified, print returns to the original site of job origin, or, in the case of interactive use, to the central site. The destination parameter is a single letter from the list of legal job source characters (see Appendix E of the *SCOPE/HUSTLER Reference Manual*), or UM or WU when disposing to a Merit Host. Note: You must be authorized for a particular source in order to dispose a file to it.

ncopies optional copies parameter; ncopies specifies the number of additional copies desired ($0 \leq n \leq 63$). The total number of cards or pages is still controlled by the authorized limit for the particular job.

lmt optional card or page limit. If not specified in interactive use, the Authorization File maximum card or page 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.)

acct a local file containing accounting information for jobs disposed to Wayne State University or the University of Michigan through the Merit Network. This parameter should only be used when disposing jobs through the Merit Network. The local file acct defaults to INPUT. A job may be disposed to the Merit Network by specifying a destination of WU or UM. See the *Merit User's Guide* for an example of disposing batch jobs via the Merit Network.

There is another form of the DISPOSE command available.

```
DISPOSE,*lfn,dis.
```

Note: This form of the DISPOSE command should be avoided in interactive use. This form does not release the file immediately, but waits for job termination or a separate RETURN request. However, the logout procedure necessary to terminate an interactive job in effect destroys the file to be disposed.

Examples:

Below are four examples of the DISPOSE command, each followed by a brief explanation.

DISPOSE,MYFILE,PAF.	prints MYFILE on 96-character ASCII upper/lower case printer at the central site.
DISPOSE,MYFILE,PR=V.	prints MYFILE at the Engineering remote batch terminal.
DISPOSE,OUT,PB,L=500.	punches OUT in binary mode and restricts the punch to 500 cards.
DISPOSE,A,PA,C=9,L=200.	prints 10 copies of A on any ASCII Fancy printer at the central site. The total number of pages for all 10 copies may not exceed 200.

Since a permanent file cannot be disposed directly, you could enter the following commands to print a copy of a permanent file on a central site ASCII printer.

```
ATTACH,IN,LONGDOCUMENT,PW=SNOOPY.
COPYSAF,IN,OUT.
DISPOSE,OUT,PA.
```

When IN is copied to OUT, each line is shifted one column to the right to ensure that column 1, the carriage control column, is blank. Note that OUT need not be rewound before the DISPOSE.

7.2 Creating Batch Input Files

In order to be disposed to the central site for processing, your job file (which may be created under EDITOR) must have the same format as a card deck that is submitted at the central site. In other words, the file must have all the features of a card deck except for the sequence card, PNC and password card. The sequence card is eliminated because the sequence number is automatically assigned, and the PNC and password card are not needed since you established your right to access by successfully logging in.

Your job card entry should have the same form as a job card submitted for batch input except that you must include a problem number parameter on it. The general form is:

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

id authorized user ID. This must be the same as the ID under which you are currently logged in.

pn your authorized problem number (6 or 7 digits) pn must be the same problem number under which you are currently logged in.

rg the rate group.

ct the estimated maximum job cost in cents.

fl the maximum field length (octal).

t the time limit in seconds.

n the number of 7-track tape units to be used (0-4).

x the number of 9-track tape units to be used (0-4).

l the page limit.

c the punched card limit.

m the maximum mass storage space in hundreds of PRUs (octal).

Parameters may appear in any order (after the ID) and the parameter list is terminated by a period. Only your ID and problem number are required.

Once the job file, lfn, has been constructed, it can be disposed to the central site for batch processing by typing

DISPOSE,lfn,IN.[1]

The sequence number that has been assigned to the job is displayed, and lfn will be released from the terminal. Note that the dis, C=ncopies, and L=lmt parameters cannot be specified with the disposition IN. You may still specify the destination, page/card limit, and number of desired copies of the output, however, by inserting a DISPOSE command (to a printer or card punch) in the control section of the job file. Otherwise, output will be printed (or punched) at the central site.

EDITOR facilitates the construction of control sections when 'SYSTEM,BATCH' is used. (See Section 3.5.2 and 3.11) In this mode, EDITOR replaces *JOBCARD* with the proper id and PNpn parameters. Also by typing GO, the EWFILE is automatically saved and then a DISPOSE command is executed.

EDITOR facilitates the construction of control sections when 'SYSTEM,BATCH' is used. (See Section 3.5.1 and 3.10.7.) In this mode, EDITOR replaces *JOBCARD* with the proper id and PNpn parameters. Also by typing GO, the EWFILE is automatically saved and then a DISPOSE command is executed.

The following examples illustrate the construction of job files and various applications of the 'DISPOSE,lfn,IN.' command. The examples have been condensed.

Example 1:

One of the simplest and most frequently used examples of submitting a batch job from an interactive terminal shows retrieval of program (or data) which is stored on a PF dump tape. In this case, you might create and DISPOSE a job file by using the following directives and commands.

```
OK-SYSTEM,BATCH.
OK-N,10.
10=*JOBCARD*,RG3,MT1.
20=PFLOAD,MT=UP1001,PFN=RECDATA.
30=SAVE,GET.
OK-DISPOSE,GET,IN.
```

Depending on the system job load, the permanent file will be available after a wait of from 5 minutes to 2 hours or more. You may type

LISTAPE. or HAL,STATUS,seqno.

to determine when the magnetic tape has been mounted and read. (See STATUS, Section 2.9.2 or LISTAPE, Section 2.9.5.)

Example 2:

Assuming that your EDITOR work file contains an error-free FORTRAN program, the following set of type-ins will cause that program to be compiled and the binary to be cataloged as a permanent file. The EWFILE, which is permanent, is returned. A batch job, which transfers the permanent file to the PF dump tape UP1001, is then created and disposed to the input queue.

[1]This form of DISPOSE requires you to be authorized for source T.

```
OK-ftn.
OK-catalog,lgo,doall.
OK-return,ewfile.
OK-system,batch,
OK-n,10.
10=*jobcard*,rs3,mt1.
20=pfdump,mt=up1001,pfn=doall.
30==go.

COMPILING MAIN

SUBMITTED UNDER SEQUENCE TB50117
```

Example 3:

You may use the interactive system solely for creating and submitting a job to batch input. In this example, notice the use of the BKSP (character delete) to delete an error in line 130 and the use of the CANCEL (line delete) to eliminate the incorrect line following line 140.

```
READY 11.42.13
system,batch,n.
100=*jobcard*,jc100.
110=ftn.
120=*eor
130=*format,fortran,130-*1
READY 11.43.48

130=program any (input,output)
140=format (*hi
*format (* hi there*)
150=print 1
160=end
170=111 lgo.
170=list f
100=*JOBCARD*,JC100.
110=FTN.
111=LGO.
120=*EOR
130=      PROGRAM ANY (INPUT,OUTPUT)
140=1     FORMAT (* HI THERE*)
150=      PRINT 1
160=      END

READY 11.46.18
save,job.

READY 11.46.38
dispose,job,in.
 SUBMITTED UNDER SEQUENCE     TB50117.

READY 11.46.55
```

Your output from this job would be printed on the central site high-speed printer, and must be picked up under sequence number TB50117. If the output were to be printed at the Engineering remote batch terminal, the following DISPOSE command might be inserted into the job file.

112=DISPOSE,OUTPUT,PR=V.

To use this command successfully, you must be authorized for job source V.

8

Front-End Control Characters and Commands

Introduction

The Front-End computer system acts as an interface between interactive terminals and the main computer. The primary purpose of the Front-End is to facilitate communication between the mainframe computer system and a wide variety of terminals and minicomputers. The terminal sends data to the Front-End which temporarily stores information for the mainframe computer in an input buffer. The input buffer may contain up to five lines of input. When the main computer can process the information, the Front-End sends the accumulated information to it. Some information (i.e. Front-End commands and control characters) are processed directly by the Front-End and are not sent to the mainframe.

Control characters are special characters used for immediate terminal control or special functions. They are processed by the Front-End only. You may redefine the function of the control characters transmitted between the Front-End and your terminal (or minicomputer) to suit the characteristics of your equipment. The Front-End also processes several commands which control a number of interactive functions. Front-End commands conform to the following rules:

1. Front-End commands must begin with the Front-End command character (fecc). This distinguishes them from SCOPE/HUSTLER commands or program input. The default character is % but the fecc may be changed using the %FECC command (Section 8.6.1).

2. They must terminate with an end-of-line character (default is carriage return, see Section 8.1.1).

3. Front-End commands must appear alone on a line, with the fecc as the first character on the line.

Front-End commands may be typed in either upper or lower case. They need not be terminated by a period, but you may supply a period if you wish. Comma and blank are both legal delimiters, similar to EDITOR syntax. Since these commands are accepted directly by the Front-End computer, they may be entered at any time, regardless of the state of your program on the MSU system. Front-End commands may also originate from the main system (see Section 8.7).

Chapter Directory

In this chapter, the Front-End commands and control characters are divided into six functional groups followed by a section describing the use of Front-End commands in programs. Summary tables of all Front-End commands and characters appear in Appendix G for easy reference.

8.2 **Terminal Attributes**

Describes terminal attributes to the Front-End to facilitate communication.

8.3 **I/O Control Functions**

Discusses commands and characters which affect the input and output of data.

8.4 **Display Information**

Describes an assortment of commands which display information at the terminal, such as job, connection and terminal status.

8.5 **Running Multiple Jobs**

Explains the commands which allow you to have more than one job running simultaneously, either at MSU or through the Merit Network.

8.6 **Redefining Front-End Control Functions**

Discusses the command which allows you to tailor your computing environment to suit your needs by redefining Front-End control characters.

8.7 **Front-End Commands from the Main System**

Explains how programs and jobs running on the main computer system can send commands to the Front-End.

8.1 Basic Editing and Program Control

The following section describes the control characters which allow you to perform basic editing of an input line at the terminal. These characters act on a line which has not yet been sent to the Front-End computer. The ABORT character is also described here. This character allows you to terminate a running program and return control to the operating system.

8.1.1 Terminating a Running Program

Default: escape
ASCII abbreviation = ESC
Keyboard equivalent = CTRL-[
ASCII code = '1B'

Function Code: ABORT

The ABORT or escape character is used to terminate a running program and halt all output from that program. Job termination messages will still be printed if they are in the output buffers. The following characters are always echoed to the terminal to acknowledge the ABORT: !, carriage return, line feed. If a line has been partially entered when the ABORT character is typed, the line is deleted and the following characters are returned to the terminal: !, carriage return, carriage return, line feed, OK-.

```
OK-help,f*catalog,
****CATALOG
Create a permanent file.
The CATALOG statement makes a local file permanent by assigning it a
permanent file name, plus optional passwo!

OK-
```

8.1.2
Terminating a Line

Default: carriage return
ASCII abbreviation = CR
Keyboard equivalent = CRTL-M
ASCII code = '0D'

Function Code: EOL

The EOL character terminates a unit record. Once an EOL character is typed, you can no longer edit the line (except for 'typed ahead' lines). You may enter input lines before the main computer is ready to process them. The lines which have been typed ahead wait in input buffers while another operation is in progress. Lines which have been entered in this fashion can be altered using the control character which retrieves typed ahead input (Section 8.3.3) or the Front-End command %DEQ (Section 8.3.4) which can be used to delete a typed-ahead line. Once a line has been sent to the main system, it can not be edited or deleted.

8.1.3
Deleting a Character

Default: backspace
ASCII abbreviation = BS
Keyboard equivalent = CTRL-H
ASCII code = '08'

Function Code: BKSP

The BKSP character causes the Front-End to logically backspace over the preceding character in the input line. Typing n consecutive BKSP characters causes the Front-End to backspace over n previously typed characters. This erases the previous n characters and allows you to correct any typing errors by typing the correct characters. A BKSP character at the beginning of a line has no effect. The character echoed to the terminal when the backspace character is typed may be altered, using the '%ECHO,BKSP = char' command. This is useful when the terminal in use does not have a physical backspace function.

If you have typed:

```
ATTABH,X,DATAFILE,PW=MARZIPAN.
```

the computer will receive:

```
ATTACH,X,DATAFILE,PW=MARZIPAN.
```

8.1.4
Deleting a Line

Default: cancel
ASCII abbreviation = CAN
Keyboard equivalent = CTRL-X
ASCII code = '18'

Function Code: CANCEL

The CANCEL character causes the line currently being typed in to be discarded. The following characters are echoed back to the terminal to signal that the line has been deleted: carriage return, \\\\, carriage return and line feed.

This character is commonly used when you are entering a line and wish to start over.

Suppose you have typed:

ATTACH,X,DATAFILE

and realize you have specified the wrong permanent file. You can delete this input line by typing the line delete character. The Front-End responds by overprinting the deleted line with \\\\ and moving on to the next line.

ATTACH,X,DATAFILE

Now you can enter the correct command on a new line.

8.1.5 Sending Control Characters as Data

Default: data link escape
ASCII abbreviation = DLE
Keyboard equivalent = CTRL-P
ASCII code = '10'

Function Code: LITERAL

The LITERAL character allows you to enter special characters as data rather than control characters. The character sent immediately after the LITERAL character is not processed as a control function. It is sent to your input buffer for transmission to the main computer. This allows you to send special characters, such as a carriage return, to your program as data without affecting terminal function.

For example, if you were creating an ASCII File, you could use LITERAL to overprint a character. To represent the line:

A. Introduction to Production Methods.

Enter:

A[LITERAL][BKSP]_. Introduction to Production Methods.

8.2 Terminal Attributes

Different terminals have different physical characteristics. Several Front-End commands exist to describe these characteristics to the Front-End computer and facilitate communication between the terminal and the 6500.

8.2.1
Default Values for the Terminal

The Front-End command, %TERMINAL, sets the appropriate default values for the control functions of your terminal. The control functions defined by this command are:

1. parity, an error checking mechanism (See also Section 8.2.2)
2. delay, which allows the terminal to move without loss of data (See also Section 8.2.3)
3. right margin, which sets the length of an output line (See also Section 8.3.8)

The command is:

%TERMINAL,termtype

where:

termtype is a terminal identifier (i.e. DEC, CRT). A list of terminal identifiers appears in Appendix B.

See section 8.2.4 for use of %TERMINAL with alternate character sets.

Default terminal control functions are determined by baud rate when you log in.

BAUD RATE	PARITY	DELAY					MARGIN
		carriage return	line feed	horizontal tab	vertical tab	form feed	
110	even	2	0	10	10	10	72
300	even	5	0	0	0	0	79
1200	even	3	2	0	0	0	80

8.2.2
The Error Checking Process

Parity checking is an error checking process which adds up the number of bits in a unit of data, computes the parity bit required and checks the calculated parity bit with the actual parity bit transmitted with the data. If the total number of '1' bits is even, parity is even. If the total number of '1' bits is odd, parity is odd.

The %PARITY command sets the criterion for the parity checking process. The format of the command is

%PARITY,{EVEN|ODD|NONE}.

The default is EVEN. %PARITY,NONE is used for binary data which has no parity bit (see Section 5.1.4). The computer ignores parity errors on input. The use of full-duplex echo printing allows you to detect transmission errors on input because the bad character will be displayed at the terminal. Full duplex echo printing causes the Front-End to print all data as it is received. (See Section 8.3.5 for a discussion of full and half-duplex transmission.)

NOTE: Even or odd parity implies that only 7 data bits are passed between the main computer and the terminal. To read 8-bit binary data requires setting PARITY to NONE in addition to using a connected file type of "BI".

8.2.3
Delaying Data Transmission

The %CDELAY Front-End command allows you to set the delay time for the indicated terminal control functions (such as carriage return, line feed), before printing is resumed. This gives the terminal time to perform the function without loss of information. This is a physical limit imposed by terminal construction. Nulls (ASCII '00') are sent to the terminal following the control function. In addition to this command, the %TERMINAL command sets default values for delay transmission according to the characteristics of the terminal.

The command format is:

%CDELAY[,CR=n_1][,LF=n_2][,HT=n_3][,VT=n_4][,FF=n_5].

where:

CR carriage return.

LF line feed.

HT horizontal tab.

VT vertical tab.

FF form feed.

n_i number of null characters sent to the terminal following the output of the control function. A different number of nulls may be set for each control function.

NOTE: Binary (BI) and ASCII (AS) output files will have no nulls added.

8.2.4
Alternate Character Sets

The purpose of the %ALTCHAR command is to tell the Front-end that your terminal needs translation from ASCII to another character set. This allows the Front-end to make intelligible the output of any programs that do not use the terminal's native character codes.

A terminal may be specified as being in an alternate character set using the Front-end command:

%ALTCHAR,{charset|OFF|NONE}[,AUTO].

where:

charset is the name of the alternate character set being used.

Currently supported alternate character sets include:

APLTYPE The typewriter pairing character set which is sent by APL under the 'TT=TYPE' parameter.

APLBIT The bit-pairing character set, sent by APL under the 'TT=BIT' parameter.

NONE "NONE" is synonymous with "OFF" and puts the terminal
OFF in the ASCII character set until the next %ALTCHAR command changes is.

AUTO indicates that the terminal can shift automatically out of and into the standard character set (ASCII) in response to the SO (shift-out,CTRL-N) or SI (shift-in,CRTL-O) character sent from the computer. The ASCII character SO causes the terminal to shift out of the standard character set, into the alternate character set. SI causes the terminal to shift back into the standard character set. If AUTO is given, the Front-end assumes that the terminal is initially in the ASCII character set. If AUTO is omitted, the Front-end assumes that the terminal is in the alternate character set and will not be shifted to ASCII. The Front-end will not interpret any SO or SI character sent to the terminal. If the terminal is later switched to the ASCII character set, the command '%ALTCHAR,OFF' must be given.

The %TERMINAL command can be used to specify terminals with alternate character sets (see Section 8.2.1).

%TERMINAL,TELAPL is equivalent to:

```
%TERMINAL,TELRAY.
%ALTCHAR,APLTYPE,AUTO.
```

%TERMINAL,DECAPL is equivalent to:

```
%TERMINAL,DEC.
%ALTCHAR,APLTYPE,AUTO.
```

Use of '%TERMINAL,TELAPL' or '%TERMINAL,DECAPL' eliminates the need to use the %ALTCHAR command.

The Front-end will recognize SO and SI only if the AUTO parameter was given on the %ALTCHAR command, or if the terminal type given in the %TERMINAL command is an automatic type (as in TELAPL or DECAPL). Otherwise SO and SI will have no effect on whether or not translation is done. You may switch the terminal manually by using the SO and SI keys with '%ECHO,ON'. '%SHOWNPC,ON' will interfere with recognition of SO and SI. SO and SI will not be interpreted on input echo if '%SHOWNPC,PART' is selected.

8.3 I/O Control Functions

This section discusses commands and control characters which affect the input and output of information.

8.3.1 Stopping and Starting Output

Default:	STOPOUT	device control 4
		ASCII abbreviation = DC4
		Keyboard equivalent = CTRL-T
		ASCII code = '14'
	HALTOUT	device control 3
		ASCII abbreviation = DC3
		Keyboard equivalent = CTRL-S
		ASCII code = '13'
	STARTOUT	device control 1
		ASCII abbreviation = DC1
		Keyboard equivalent = CTRL-Q
		ASCII code = '11'

Entering a STOPOUT character causes output to halt at the end of the current line. Entering a HALTOUT character causes output to halt immediately. Both the HALTOUT and STOPOUT characters will suspend output even if output has not yet begun.

The STARTOUT character causes output to resume. CANCEL (see Section 8.1.4) and ABORT (see Section 8.1.1) will also cause output to resume.

8.3.2
Terminating the Display of an Output Line

Default: substitute
ASCII abbreviation = SUB
Keyboard equivalent = CTRL-Z
ASCII code = '1A'

Function Code: TERMOUT

Entering the TERMOUT character discards the remainder of the output line in progress. This has no effect on anything but the current output line. The next output line, if any, is immediately begun.

8.3.3
Retrieving a Previous Input Line

Default: negative acknowledge
ASCII abbreviation = NAK
Keyboard equivalent = CTRL-U
ASCII code = '15'

Function Code: RETIN

Entering the RETIN character causes the current input line to be discarded, replacing it with the previously entered input line if that line has not yet been read by the main system. You can repeat this process as long as there are 'typed ahead' input lines in your input buffer. The retrieved line will be printed; at this point it can be edited or deleted.

Suppose you have typed ahead five lines, which are waiting in the input buffer and decide to retrieve the last line for correction.

```
ATTACH,DATA,ASSIGNMENT1.
SKIPF,DATA.
COPYCR,DATA,NEWDATA,2.
SKIPF,DATA,2.
COPYCR,DATA,NEWDATA,2.
```

Entering the RETIN control character will retrieve the last line in the buffer, making it the current input line. You can proceed to edit the line as if the EOL character had not been sent.

```
COPYCR,DATA,NEWDATA,2.
```

This correction results in the command

```
COPYCR,DATA,NEWDATA,3.
```

being sent to the Front-End, where it will wait in the input buffer until the main system is ready to process it.

The RETIN character is ignored if typed while output is in progress, or if there is no typed ahead input. (That is, if the main system has read the last line sent.) If you discard the current input line, this is indicated by a carriage return followed by \\\\. Then the retrieved input line is printed.

8.3.4 Deleting Lines from the Input Buffer

The %DEQ command allows you to delete input lines which are waiting in the input buffer. The lines are deleted on a last in /first out basis.

The format of the command is:

```
%DEQ,n[,LIST]
```

where:

n is an integer indicating the number of lines to be deleted. If n is omitted, n=1 is assumed.

LIST causes the lines to be printed as they are deleted.

Suppose your input buffer contains the same five lines used in the example in Section 8.3.3. Typing:

```
%DEQ,4.
```

deletes the last four lines in the input buffer. Your input buffer now contains:

```
ATTACH,DATA,ASSIGNMENT1.
```

8.3.5 Terminal Communication: Full/Half Duplex

Terminals can communicate with the computer in two modes: full-duplex or half-duplex. Full-duplex communication permits signals to be both sent and received simultaneously by the computer and the terminal. Half-duplex allows communication in one direction at a time between the terminal and the computer. When the terminal is in full-duplex mode, characters are not printed at the terminal as they are entered; they are returned by the Front-End as they were received (this is called echoing). This allows you to verify that the correct character was indeed received by the computer. Under half-duplex mode, characters are printed by the terminal as they are entered.

The %ECHO command controls the transmission of characters from the Front-End to the terminal.

The ECHO command has two forms:

```
%ECHO,{ON|OFF}
%ECHO,BKSP=char
```

where:

ON causes input characters to be returned to the terminal by the Front-End as they are typed. ECHO ON is intended for full-duplex terminal use. This gives you an opportunity to ensure that the correct character was transmitted. Terminal control characters, with the exception of EOL and BKSP, are not echo printed.

OFF is intended for half-duplex use of a terminal. The Front-End echo prints no input characters to the terminal.

BKSP=char changes the character echoed by the BKSP function. This is useful when you are using a terminal which does not have a physical backspace function.

%ECHO,OFF is the default setting for 110 baud terminals while %ECHO,ON is the default setting for terminals faster than 110 baud.

A problem can occur when you are entering information at a terminal with the full/half duplex switch set on half when ECHO is ON. Every character you type will be printed twice; once by the terminal and again by the Front-End. This is obvious at login.

```
01/16/81  MSU HUSTLER 2        LSD 50.26  01/13/81  CYBER750

TYPE PASSWORD, PN, AND USER ID.
■■■■■■■■■■aarrkk,,00111122660033,,mmuurrpphhyy
```

In this case, you can correct the situation by changing the full/half duplex setting to full or by typing:

%ECHO,OFF.

If ECHO is ON and you enter a character while the Front-End is sending normal output to the terminal: the input is held until the terminal completes the current output line, then the input line is echoed.

8.3.6
Exchanging Communication Mode: Full/Half Duplex

Default: synchronous idle
ASCII abbreviation=SYN
Keyboard equivalent=CTRL-V
ASCII code='16'

Function Code: ECHO

The ECHO character changes the transmission characteristics of the terminal. If the terminal is in ECHO,ON mode (see Section 8.3.5) entering the switch echoback character will reverse full duplex with echo printing to full duplex without echo printing or vice-versa. This is useful when you wish to enter your password without having it echoed at the terminal. Echoing will be resumed after an EOL, CANCEL or ABORT.

If the terminal is in ECHO,OFF mode the ECHO character is ignored.

8.3.7 Displaying an Input Line

Default: acknowledge
ASCII abbreviation = ACK
Keyboard equivalent = CTRL-F
ASCII code = '06'

Function Code: LECHO

If you have made several corrections to an input line, the contents of the line may be obscured by the corrections. Entering the LECHO character allows you to examine the contents of the line and make further additions or corrections. Upon receipt of this control character, a carriage return and line feed are sent to the terminal followed by the contents of the line.

Suppose the following line has been typed in, but the end-of-line character has not yet been entered.

```
READY 11.34.23
CATALOG,EWFILE,ARRFILE,ID=ABELL,RP=999,TK=ROUSE.
```

Entering the LECHO character causes the Front-End to return the contents of the line for verification.

```
CATALOG,EWFILE,ARRFILE,ID=ABELL,RP=999,TK=ROUSE.
```

Additional editing can be performed before the EOL character is typed.

If an LECHO character is entered while output is in progress, the character is ignored.

8.3.8 Setting Maximum Output Line Length

The %RMARGIN command resets the maximum length of an output line. The format of the command is:

```
%RMARGIN,cwidth
```

where:

cwidth is the number of characters in the output line. The default setting is the number of characters accommodated by the carriage width of the terminal. If cwidth is 0, output lines will not be folded by RMARGIN processing. This may result in overprinting if your terminal does not fold lines automatically.

If an output record is larger than cwidth, the output line is split after cwidth characters and continued on the next physical line. %RMARGIN has no effect for BI (binary) and AS (ASCII) files.

Suppose you wish to restrict the length of an output line to sixty characters. Enter:

```
%RMARGIN,60
```

Any records greater than sixty characters in length will be continued on the next line. For example the following data record:

```
This is a data record containing more than sixty characters of sample information. EOL
```

would be displayed at the terminal as follows:

```
This is a data record containing more than sixty characters
of sample information.
```

8.3.9 Setting Maximum Input Line Length

The %INLEN command specifies the maximum length of an input line in number of characters. The default value is 240 characters.

The format of the command is:

```
%INLEN,n
```

where $0 \leqslant n \leqslant 240$. If $n = 0$, the input line length is set to 240 characters.

An input record is terminated when:

1. You enter an EOL character, or
2. n characters have been entered,

whichever comes first. If n characters have been entered, the current line is terminated, and the next character input becomes the first character of the next line. INLEN is ignored for Front-End commands (i.e. a Front-End command may always contain up to 240 characters).

8.3.10 Entering Lines Longer than the Terminal Width

Default: end of block
ASCII abbreviation = ETB
Keyboard equivalent = CTRL-W
ASCII code = '17'

Function Code: CONT

The CONT character allows you to enter a line which exceeds the terminal width without causing overprinting. This character causes a carriage return and line feed to be returned to the terminal without entering the carriage return and line feed in the input line.

If you were working on a CRT with a screen width of 72 characters and wanted to enter a record longer than 72 characters, use the CONT character.

```
This is a data record containing more than 72 characters of sample[CONT]
information[EOL]
```

The CONT character may be entered at any point. Remember that you cannot enter a line greater than the maximum input line length (default=240 characters).

You can also use the CONT character to move to a new line in order to enter input ahead of the running program which has not yet prompted for that input.

8.3.11 Displaying Non-printing Characters

The command, %SHOWNPC, allows you to print graphic representations of non-printing characters at the terminal. %SHOWNPC stands for "show non-printing characters." This command is compatible with the %NPC command of Hermes, the Merit Network Front-End computer. On the MSU system, %NPC is a synonym of %SHOWNPC for the sake of compatibility with Hermes.

The format of the command is:

```
%[SHOW]NPC[,OFF|PART|ON][,{MNEM|CTRL|char}]
```

where:

OFF — specifies that non-printing characters not be displayed. This is the default.

PART — displays a subset of all control codes. Codes that produce a visible or audible action by the terminal are excluded. If %SHOWNPC is typed alone, PART is assumed.

ON — requests that all control codes be graphically displayed at the terminal.

The remaining parameters specify the way control codes will be displayed.

MNEM — specifies that control codes be displayed using the standard ASCII mnemonic symbol in brackets (e.g. [SOH], [STX], [CAN], etc.).

CTRL — requests that control codes be displayed using the form "<CTRL-n>" (except for DEL which is shown as <RUBOUT>).

char — requests that control codes be indicated by a character chosen by the user. It may be ASCII BEL or any printable character except blank or comma. The default is %.

When you first log in, %SHOWNPC is OFF. Typing "%SHOWNPC" is equivalent to typing

```
%SHOWNPC,PART,%
```

8.3.12
Using a Carriage Control Character

The %CCTL command controls the interpretation of the first character of each line of output to the terminal.

The format of the command is:

%CCTL,{ON|OFF}

where:

ON causes carriage control characters to be interpreted as specified for the character set currently in use.

OFF causes all lines to be single-spaced and prints the first character of every line.

The default is "%CCTL,ON," which allows carriage controls to be processed for output from DC (Display Code), AF (ASCII Fancy) and BF (Binary Fancy) files. BI (Binary) and AS (ASCII) files do not have carriage controls.

This can be useful for debugging. You can display the contents of an ASCII file using WRITEPT (see Section 2.9.4), but you must set CCTL to OFF to see the contents of the first column of the file.

8.3.13
Reading Paper Tape

The %READER,ON command is used with paper tape (or other forms of auxiliary input). It allows the Front-End computer to automatically stop and start input. When the input buffer in the Front-End computer can no longer hold additional input, the Front-End halts input by transmitting a DC3 character to the terminal. Input is resumed as soon as the input buffer empties and the Front-End transmits a DC1 character to the terminal. This ensures that information will not be lost.

The format of the command is:

%READER,{ON|OFF}

You set %READER,ON in conjunction with the READPT command (Section 2.8.1) or with a running program reading from a connected file. This is not necessary when using TPREAD (see Section 2.8.1). The TPREAD command automatically sends a %READER,ON command to the Front-End. When TPREAD is finished, %READER,OFF is sent to the Front-End.

8.3.14
Reading Binary Data

If you wish to read binary data, it is necessary to set %BINARY,ON. This treats all special characters as data. All characters are sent as literals. The text cannot be edited because all editing control characters and commands will be sent as data instead of being processed normally. You terminate the transmission of binary data by depressing the BREAK key. This resets %BINARY to OFF.

The format of the command is:

%BINARY,{ON|OFF}

The default is %BINARY,OFF. If %BINARY is entered with no parameters, %BINARY is set to ON.

8.4 Displaying Information

You can display information about your job, connection or the system in general using the following commands.

8.4.1 Displaying Job Status

The %JOBSTAT command returns the current status of your job on the main system. The command is:

%JOBSTAT

The message you receive from the Front-End has the following format:

state CM=words, CP=secs dayfile

where:

state	is a message describing the current job state
words	current (or last used) job field length in octal
secs	CPU time elapsed since the beginning of a session in decimal seconds
dayfile	first twenty characters of the last job dayfile message

If you are connected to the Merit Network, an attempt to use the %JOBSTAT command will return an error message.

The following is an example of a %JOBSTAT message:

IDLE CM= 2200,CP=174.8 CYCLE SUCCESSFULLY P

8.4.2 Displaying Connection Information

The %CONSTAT command prints out details of your connection with the Front-End computer. This information can be useful to the consultants in tracing possible terminal problems. Entering:

%CONSTAT

causes the Front-End to print:

SOCKET=ss, PORT A=aa, PORT B=bb

where:

ss the socket number which corresponds to the interactive phone line.

aa the port number in the Front-End computer of the user's primary connection. A zero value for a port number means that no connection exists.

bb the port number in the Front-End computer of the user's secondary connection.

ss, aa and bb are decimal numbers.

The following is an example of a %CONSTAT message:

SOCKET= 4,PORT A= 10, PORT B= 0

8.4.3 Displaying Time

Entering the %TIME command causes the current time and date to be printed at your terminal. The command is:

%TIME

The Front-End responds with:

hh:mm:ss mm/dd/yy

where:

hh:mm:ss is the current time in hours, minutes and seconds. NOTE: This time is maintained by the Front-End and may differ slightly from the time maintained by the mainframe.

mm/dd/yy is the current date.

For example:

14:21:01 11/13/77

8.4.4 Displaying Terminal Attributes

The %TERMSTAT command displays current terminal attributes. When you enter:

%TERMSTAT

the Front-End responds with the message:

```
RMARGIN= n, INLEN= n
CR= n, LF= n, HT= n, VT= n, FF= n
```

```
PARITY= parity, TERMINAL= type
CCTL= cctl, READER= reader, MIX= mix
SHOWNPC= display,mode
BACKSPACE ECHO= char
ALTERNATE CHARACTER SET= alternate set
FECC= char
```

All numbers are decimal.

For example:

```
RMARGIN=  0,INLEN= 240
CR=  5,LF=  0,HT=  0,VT=  0,FF=  0
PARITY= EVEN,TERMINAL= TI700
CCTL= ON,READER= OFF,MIX= OFF
SHOWNPC= PART,MNEM
BACKSPACE ECHO= [BS]/CTRL-H
ALTERNATE CHARACTER SET= NONE
FECC= %
```

8.4.5 Displaying Front-End Status

The %FESTAT command displays the number of interactive jobs currently on the system. Typing:

```
%FESTAT
```

causes the Front-End to respond with:

```
LINES = m, MISTIC2 = n, MERIT = r
110=n1, 300=n3, 1200=n12
```

where:

m is the number of interactive lines connected to the Front-End.

n is the number of interactive connections to the mainframe.

r is the number of interactive connections to the Merit Network.

n_i is the number of interactive connections at the corresponding baud rate.

All numbers are decimal.

For example:

```
READY 17.28.18
%festat
 LINES=  4,MISTIC2=  4,MERIT=  0
 (110=  0, 300=  2, 1200=  2)
```

8.4.6
Displaying the Log-in Message

The %LOGINMSG command allows you to display the log-in message at the terminal at any time. The command is:

%LOGINMSG

The Front-End responds with the current log-in message, for example:

```
%loginmsg
DOWN 03:45, FRIDAY, UP 08:00.
```

8.5
Running Multiple Jobs.

Two separate jobs can be connected to a single terminal. You can run both jobs at MSU, one at MSU and the other through the Merit Network, or both on the network.

There are several Front-End commands which control the communication between the terminal and the jobs.

8.5.1
Obtaining a Second MSU Connection

The %LOGIN command establishes another connection with the MSU computer system. Entering:

%LOGIN

generates the message:

[PORT n]

where:

n is the port number on the MSU computer system.

At this point, you are prompted for the standard log-in information. You must log in using a different id or account; otherwise you will receive a message stating that you are already logged in.

NOTE: Use of %LOGIN will be restricted to prevent overloading of the system.

```
%login

[PORT  89]


01/16/81  MSU HUSTLER 2       LSD 50.26  01/13/81  CYBER750

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

8.5.2
Obtaining a Merit Network Connection

The %NETCNT command establishes an interactive connection through the Merit Network.

The format of the command is:

%NETCNT,[UM|WU]

where:

UM establishes a connection to the University of Michigan.
WU establishes a connection to Wayne State University.

You are then prompted to log in under whichever host was requested.

```
.netcnt,um

[PORT  86]

   MTS : Ann Arbor (MN39-00430)
#
```

8.5.3
Switching Input Transmission between Two Connections

The %FLIP command determines which interactive job is in direct communication with your terminal. The job in direct communication with the terminal is called the primary connection. The other job is called the secondary connection.

The command is entered:

%FLIP

This causes the primary and secondary connections to be exchanged. The Front-End prints the port number of the current primary connection.

All input from the terminal goes to the current primary connection. Output transmission is controlled by the %MIX command.

The %MSU and %NET commands are special cases of %FLIP which are used in conjunction with the Merit Network.

%MSU establishes the MSU connection as the primary connection.

%NET establishes the network connection as the primary connection.

8.5.4
Switching Output Transmission between Two Connections

The %MIX command controls the transmission of output from multiple jobs to your terminal. The format of the command is:

%MIX,{ON|OFF}

where:

OFF All output from the primary connection is sent to the terminal. All output from the secondary connection is held until you enter the %FLIP command to exchange connections. The job running on your secondary connection will stop running if the output buffers fill up.

ON allows output from both connections to be intermixed at the terminal on a line by line basis. This is the default condition.

8.5.5
Terminating Your Primary Connection

The %QUIT command disconnects your primary connection. The secondary connection becomes the current primary connection. If you had only one connection, you are disconnected. This is the same as hanging up the phone without logging out.

The command is entered:

%QUIT

8.6
Redefining Front-End Control Functions

One of the useful features of the Front-End computer system is that it allows you to redefine the functions of special characters. This gives the system greater flexibility in dealing with the peculiarities of terminals and minicomputers. The commands which perform the redefinitions are described in the following sections.

8.6.1
Redefining the Front-End Command Character

The default value for the Front-End command character is the percent sign (%), but you may redefine this character if it is more convenient to do so. The command is:

%FECC,{x|abb|CTRLn|DATA}

where:

x is a single ASCII character.

abb is a standard ASCII abbreviation for a control function, such as 'SOH'. See Table 8.1.

CTRLn — the letters 'CTRL' followed by a single letter (A-Z). See Table 8.1.

DATA — the character is to be passed as data only: no control function is associated with it. Until you redefine a Front-End control character, there will be no fecc. The fecc can be reinstated using the SCOPE/HUSTLER command 'FECMD,FECC,char', which is described in Section 8.7.1.

8.6.2 Redefining a Control Character

You can redefine any control character. Characters are only affected on input from a terminal. This allows users with nonstandard terminals a greater degree of flexibility. The %ALTER command will change all control characters except the Front-End command character.

The %ALTER command has three forms:

1. %ALTER,char=function[,char=function,...]

where:

char — is the control character whose function is altered. char must be a standard ASCII abbreviation or the letters 'CTRL' followed by a single letter (A-Z) or a single control character preceded by a LITERAL control character. See Table 8.1.

function — is the function which the control character is to perform. See Table 8.2.

If you entered:

```
%ALTER,CTRLA=BKSP
```

The CTRL-A (SOH) key would function as BKSP. BS (backspace) will also function to delete characters because it has not been altered. To disable BS you would enter:

```
%ALTER,BS=DATA
```

2. %ALTER,RESET

This command resets all control characters to their default values.

3. %ALTER,LIST

This causes all control characters and their functions to be listed at the terminal. Control characters which are treated as DATA are not listed.

Table 8.1
Standard ASCII Mnemonics and Keyboard Equivalents

Standard ASCII Abbreviation	Keyboard Equivalent	Standard ASCII Abbreviation	Keyboard Equivalent
NUL	NULL	DLE	CTRLP
SOH	CTRLA	DC1	CTRLQ
STX	CTRLB	DC2	CTRLR
ETX	CTRLC	DC3	CTRLS
EOT	CTRLD	DC4	CTRLT
ENQ	CTRLE	NAK	CTRLU
ACK	CTRLF	SYN	CTRLV
BEL	CTRLG BELL	ETB	CTRLW
BS	CTRLH	CAN	CTRLX
HT	CTRLI	EM	CTRLY
LF	CTRLJ	SUB	CTRLZ
VT	CTRLK	ESC	
FF	CTRLL	FS	
CR	CTRLM	GS	
SO	CTRLN	RS	
SI	CTRLO	US	
		DEL	

Table 8.2

Function Code	Valid Functions
DATA	the character is to be passed as data only: no control function is associated with it.
EOL	end of line
BKSP	character delete
CANCEL	line delete
ABORT	job abort or end of tape
ECHO	switch echoback
IGNORE	echo and discard the character
LITERAL	literal next
LECHO	line echo
TERMOUT	terminate output line
STOPOUT	suspend output at end of current line
HALTOUT	halt output immediately
STARTOUT	resume output
RETIN	retrieve input
CONT	line continue

8.7
Front-End Commands from the Main Computer System

Programs and jobs on the main system may send commands to the Front-End computer. When a program issues a Front-End command, the Front-End computer receives the command and sends a reply to the job or program, indicating the success or failure of the command. No indication of error is automatically returned to your terminal. Commands which would normally cause output to appear at a terminal, such as %CONSTAT or %TIME, still generate output at the terminal.

8.7.1
Sending Front-End Commands from an Exec File

To send a Front-End command from a job control section or from an EXEC file, use the control statement:

```
FECMD,command.
```

where:

command — is any valid Front-End command with the Front-End command character (%) omitted. Note that % cannot be referenced by FECMD. FECMD is a Display code control statement. % is not part of the Display code character set.[1]

If the command is in error, the job will abort and the appropriate error message will be printed.

[1]If you need to reset the FECC from DATA to %, you must do so indirectly. For example:

```
FECMD,FECC,*
*FECC,%
```

Note that the ASCII character (%) is not used with the FECMD command. A Display code character (*) is used instead.

8.7.2
Sending Front-End Commands from FORTRAN Programs

To Send a Front-end Command Using FECMD.

To send a Front-End command from a running FORTRAN program, call the function subprogram, FECMD. The calling sequence for FECMD is:

```
X=FECMD(command[,cc])
```

where:

command — is the Front-End command in Display code without the Front-End command character (%) and terminated by a zero byte: it may be the name of an array containing the command or the command itself in a Hollerith string.

cc — character code alone or in 2L or 2H format: cc may be specified DC, BI, AS, AF, 2HDC, 2HBI, 2HAS, 2HAF, 2LDC, 2LBI, 2LAS, or 2LAF. The default is 2LDC.

X — is an error code. If the Front-End detects an error in the command, the Front-End returns X≠0. If the command is valid, the Front-End returns X=0. Since FECMD is, by FORTRAN convention, a real function, X will be returned as a real variable.

For example:

```
INTEGER COMMAND(3)
DATA COMMAND/15HALTER,CTRLA=EOL,0/
.
.
.
X=FECMD(COMMAND)
IF(X.NE.0)PRINT*,"BAD FE COMMAND"
```

or

```
X=FECMD('ALTER,CTRLA=EOL')
IF(X.NE.0)PRINT*,"BAD FE COMMAND"
```

Note that the default character code for the Front-End command is DC.

To request Front-end Data Display Using FEDATA.

FEDATA is a routine that will cause Front-end information to be placed in a buffer in your field length. You may access FEDATA via an FTN callable subroutine and a COMPASS macro (see Section 8.7.3).

FEDATA may be called using the FTN command:

```
CALL FEDATA(BLOCK,LENGTH)
```

where:

BLOCK is an array to hold the return data.

LENGTH is the block length in words. A length of 0 is the same as 1. If omitted, a length of 1 is assumed.

Note: the "LENGTH" parameter is optional.

The FEDATA call includes the address of a parameter block which may be one or more words long. The maximum useful length is 15 words, decimal. FEDATA will fill this block with Front-end information, based on the block length specified. See Table 8.3. If the block length is longer than 15 words, FEDATA will set the excess words to zero.

8.7.3 Sending Front-End Commands from COMPASS Programs

To Send a Front-end Command Using FECMD.

To send a Front-End command from a running COMPASS program, call the macro, FECMD. The macro has one argument which is the address of a parameter word. The command must immediately follow the parameter word. (Note that the fecc is omitted.) The form of a general macro call and parameter word is as follows:

```
      FECMD  parm
      .
      .
      .
parm  VFD    24/0,12/len,12/cc,12/0
```

where:

len is the length of the Front-End command in words.

cc character code for the character set of the command. Character sets are described in Section 5.1. This may be:

DC	Display Code
AS	ASCII
AF	ASCII Fancy
BI	Binary

For example:

```
        FECMD  MSG
        .
        .
        .
MSG     VFD    24/0,12/3,12/2HOM,12/0
        DATA   15HALTER,CTRLA=EOL
        DATA   0
```

After the command has been processed, a reply is returned in the parameter word as follows:

24/0,12/len,12/cc,11/ec,1/1

where:

len command length

cc character code, as above.

ec is an error code.
ec=0 no error
ec≠0 invalid Front-End command. See Appendix D for a list of error codes.

To Request Front-end Data Display Using FEDATA.

FEDATA is a routine that will cause Front-end information to be placed in a buffer in your field length. You may access FEDATA via an FTN callable subroutine (see Section 8.7.2) and a COMPASS macro.

FEDATA may also be called using the COMPASS macro:

LOC FEDATA BLOCK,LENGTH,RECALL

The only required parameter is the block address. If length is omitted, the macro assumes the length is pre-set in the block and the complete bit is already cleared. The block address and length parameters may be register names. A length of 0 will be treated the same as length=1.

The FEBLOK macro may be used to reserve a block of words for the FEDATA macro:

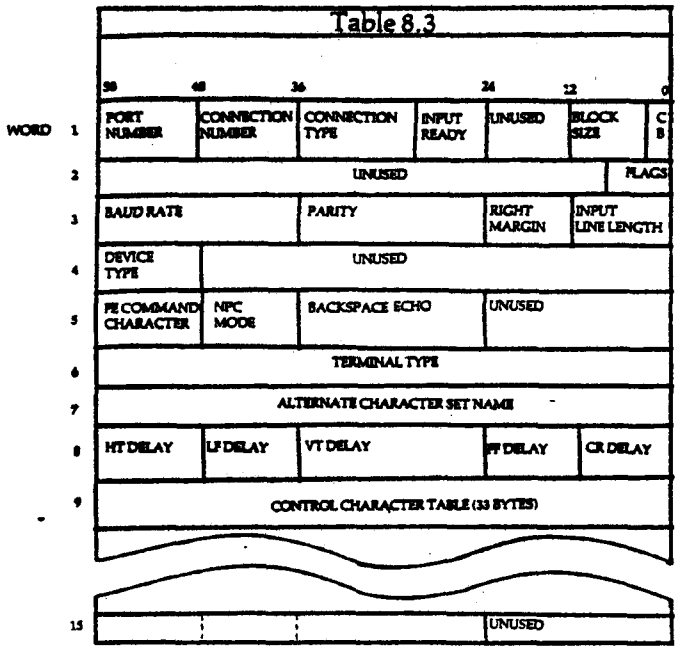
BLOCKNAME FEBLOK LENGTH

Reserves 'LENGTH' words. If length is omitted, fifteen words are reserved. The length of the field is preset in the block and the complete bit cleared.

The format of the block is given in the following table.

FEDATA RETURN BLOCK

Table 8.3



The first word of the table contains information from the Front-end port. The remaining words are taken from the socket. On in-bound Merit connections there is no socket, the connection in the Front-end is port-to-port. In this case, only the first word contains any information; the rest of the block is set to zero.

All unused fields in the block are set to zero.

Port number	the number of the Front-end port assigned on log-in; never changes.
Connection type	0 = not connected (batch or phone just hung up) 1 = Port-to-port 2 = Port-to-socket
Connection number	the number of the port or socket you are connected to, depending on connection type. (Zero if not connected.)

INPUT READY	1 = input line in the Front-end ready to be read.
Block Size	the length of the FEDATA return block in words. This is a positive integer whose values are from 1 to 15 (decimal) A value of 1 will be treated as 1. If length is greater than 15, all words after the 15th will be set to 0.
CB	FEDATA request complete bit.
Flags	Bit The bit is a "one" when the condition is true. 0 = Reader on ('%READER') 1 = Mix on ('%MIX') 2 = Carriage controls off ('%CCTL') 3 = NPC on or part ('%SHOWNPC') 4 = NPC on 5 = Binary input mode ('%BINARY') 6 = Echo off ('%ECHO') 7 = I/O Commander terminal 8 = Alternate Character set in use 9 = Enable automatic character set ('%ALT-CHAR')
Baud Rate	Current terminal speed setting as a binary integer; 110, 300, 1200 and so on.
Parity	0 = none 2 = odd 3 = even (set by '%PARITY' command)
Right Margin	0 = no right margin enabled 1-237 right margin value. (set by '%RMARGIN' command)
Input line	1-240 maximum length input line (set by '%INLEN' command)
Device Type	2 = Console TTY 3 = Interactive user terminal
FE Command character	The ASCII character used as a prefix for Front-end commands. (set by '%FECC' command)
NPC Mode	0 = interpret control characters with ASCII mnemonics 1 = interpret as <CTRL-x> Other = use this value as a single ASCII character (set by '%SHOWNPC' or '%NPC' command)

Backspace Echo	for terminals with no physical backspace (set by '%ECHO,BKSP=char')
Terminal type	Display code, left justified blank filled. See Appendix B for list. (set by '%TERMINAL' command)
Alternate Char Set Name	Display code ("H" format) (set by '%ALTCHAR' command)
Delays	Each byte contains the number of nulls that will be sent for delay after the indicated character (set by '%TERMINAL' or '%DELAY" command)
Control Char Table	1 byte per control character and underscore. (set by '%ALTER' command) The value of each byte is:

0 = literal data (DATA)
1 = EOL character (EOL)
2 = literal next (LITERAL)
3 = suspend output (STOPOUT)
4 = toggle echoback (ECHO)
5 = resume output (STARTOUT)
6 = cancel current input line (CANCEL)
7 = escape (ABORT)
8 = backspace (BKSP)
9 = echo current input buffer (LECHO)
10 = terminate current output line (TERMOUT)
11 = line continue (CONT)
12 = line dequeue (RETIN)
13 = stop output immediately (HALTOUT)
14 = ignore character (IGNORE)

Note that most of these fields can be altered by Front-end commands, and some are altered by the Front-end internally.

9

Exec Files

9.1 Exec Files

It is often necessary to repeatedly execute a series of system commands and editing directives. This can be made simpler by creating an "exec file." Exec files contain control card images that are to be executed via the EXEC command (see below) or the GO directive (see Section 3.12).

Each group of EDITOR directives in an exec file must be preceded by the command 'EDITOR.' and followed by the command 'END.'. Groups of the system commands N, OK, READY, TAPE, TAPEC, READPT, and TPREAD, must be preceded by the command 'MISTIC.' and terminated by the command 'END.'. In addition, the commands TAPE and TAPEC must be followed by an 'EDITOR.' ... 'END.' sequence. Unlike batch, the EDITOR command, when used interactively, will ignore all parameters.

Each command or directive must appear as a separate control card image in an exec file; that is, only one command per line is allowed.

The following sequence is an example of a proper exec file.

```
EDITOR.
SAVE,SET,NS.                    } EDITOR directives
END.
RETURN,OLDPL.
ATTACH,OLDPL,MYOLDPL.
RETURN,NEWPL.
UPDATE,N,I=SET,O=OUT.           } SCOPE/HUSTLER commands
CATALOG,NEWPL,MYNEWPL.
FTN,I=COMPILE,O=OUT.
DISPOSE,OUT,P2.
MISTIC.
READY.                          } Interactive system commands
END.
```

Note: The terminating character (period or right parenthesis) must be present for all commands given in this form.

In the example, the user has a FORTRAN program stored on the permanent file MYOLDPL. To update the program, place UPDATE directives in an EWFILE, then type the command

EXEC,lfn.

where lfn is the name of the local file containing the exec file (see Section 9.1.1).

The exec file is then executed: EWFILE is written to a coded file using SAVE (see Section 3.8.2); UPDATE is executed, the new PL is cataloged, then the FTN compiler compiles the new program; output is disposed to the line printer at the central site.

9.1.1
Creating an Exec File

Exec files may be created and changed using EDITOR. The example in the previous section was created as follows:

```
OK-SYSTEM GENERAL.
```

SYSTEM GENERAL or SYSTEM TEXT should be used to prevent reformatting of the commands (See Section 3.5)

```
OK-N.
```

Starts automatic line numbering. (See Section 3.7.1)

```
100 -EDITOR.
110 -SAVE,SET,NS.
120 -END.
130 -RETURN,OLDPL.
140 -ATTACH,OLDPL,MYOLDPL.
150 -RETURN,NEWPL.
160 -UPDATE,N,I -SET,O -OUT.
170 -CATALOG,NEWPL,MYNEWPL.
180 -FTN,I -COMPILE,O -OUT.
190 -DISPOSE,OUT,P2.
200 -MISTIC.
210 -READY.
220 -END.
230 -
EON-PROCESSING TEXT
OK-SAVE,EXECFIL,NS.
```

Writes EWFILE into coded file, suitable for use with EXEC.

```
OK-CATALOG,EXECFIL,MYEXECFILE.
```

This is useful if the exec file is to be used in future sessions.

9.2
Automatic Execution of Exec Files or Programs

The automatic execution procedure allows a PN manager to set up a program or an exec file that will automatically execute whenever a user under the same problem number logs in. This is useful when a program or sequence of commands is written for a novice or non-technical user; it may also be used to restrict the use of computer resources to some specific purpose. The auto-execution feature can control an entire interactive session, or it may control only the initial part of the session. Execution of the program or exec file can be made optional or mandatory by the PN manager; once this decision is made it affects all users of the problem number, and may be overridden only by the PN manager.

9.2.1
The Initialization File

The program or sequence of commands to be executed at log-in time is called the "initialization file." The initialization file 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 is done by the AUTHORF directive:

CHANGE IINIT TO LFN=lfn,PW=password,control option.

LFN=lfn — The name of the local file that contains the command sequence or program binary (i.e. relocatable, absolute overlay or segment formats) to execute at log-in time. This file is copied and cataloged with the permanent file name "INITFILEFORPNnnnnnINTERACTIVE," where nnnnn is the problem number. (Note that five digits are always used for the problem number, and the department code is ignored.) Any previous file by this name is purged.

PW=password — The turnkey password (1-9 characters) on the permanent file. The specification of a password allows the initialization permanent file to be attached, with all permissions, at times other than at the start of an interactive job. If "PW" is given alone, the user is prompted for input.

If no password is specified, the system generates a random password, which will prevent the PN manager from attaching the file except upon log-in. The PN manager can change this situation by changing the password:

AUTHORF,CHANGE IINIT PW=newpw

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

control option — This specifies whether execution of the initialization file will be optional or required. The option in effect is not changed when the file or password is modified. The control option is one of the following:

NORMAL — This selects the use of the initialization file unless explicitly suppressed by the user by the log-in option NOINIT (see Section 9.2.3). This is the default.

REQUIRED — This causes the initialization file use to be required at the start of every job. Only the PN manager may suppress execution under the REQUIRED option; an attempt by an individual user to suppress the use of the initialization file will prevent the user from logging in.

OPTIONAL — The initialization file will normally not be executed. The user may request it via the log-in option INIT (see Section 9.2.3).

The PN manager can change the initialization file simply by entering the statement:

AUTHORF,CHANGE IINIT LFN=lfn,PW=pw

where lfn is the file containing the new initialization file. The current initialization file is purged and lfn is cataloged in its place.

The control option can be changed by entering

AUTHORF,CHANGE IINIT TO control option

For example, to change the execution of the initialization file from REQUIRED to NORMAL, the following statement would be entered:

```
AUTHORF,CHANGE IINIT TO NORMAL.
```

Such a change takes effect immediately.

Changing Initialization File Status

The PN manager may change the status of the initialization file with this AUTHORF statement:

If the PN manager wishes to change the status of the initialization file, he/she may do so with this AUTHORF statement:

```
AUTHORF,CHANGE IINIT TO {OFF|ON|PURGE}.
```

OFF Discontinue use of the initialization file. The password and control option are not changed, nor is the permanent file purged. If no initialization file exists, OFF has no effect.

ON Restore use of the initialization file without changing the control option, the password, or the file. If no initialization file currently exists, specifying ON will cause a fatal error.

PURGE Terminate use of the initialization file. This also causes the permanent file to be purged, and the password and control option in the Authorization File to be cleared. If no initialization file exists, PURGE has no effect.

The control options (NORMAL, OPTIONAL, REQUIRED) and the status options (OFF, ON, PURGE) are conflicting, and may not be specified in the same AUTHORF statement.

The AUTHORF options VETO and LIST may be used with the AUTHORF statements in this section to verify initialization file changes; for a description of these options, see Section 2.7.1.

No Authorization File changes except those described in this section may be made in an 'AUTHORF,CHANGE IINIT' statement. For example, the following statement is illegal:

```
AUTHORF,CHANGE CM TO 50000,IINIT TO REQUIRED.
```

9.2.2 Displaying Initialization File Options

The PN manager can use AUTHORF to display the options in effect for the interactive initialization file.

```
AUTHORF DISPLAY IINIT
```

The value displayed is one of the following:

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

9.2.3
Requesting or Suppressing the Initialization File

If the PN manager has set the execution of the initialization file to NORMAL, the user may suppress execution upon log-in. If the PN manager set OPTIONAL, the initialization file will be executed only if the user specifically requests it. The methods for requesting or suppressing initialization when logging in are as follows:

password, problem number, user id,INIT

or

password, problem number, user id, NOINIT

NOINIT If specified, the initialization file will not be executed.

If the REQUIRED option was specified by the PN manager, the user may not use the NOINIT option; doing so will cause the log-in attempt to fail. Only the PN manager may override the REQUIRED option by giving the NOINIT parameter.

INIT This causes the initialization file to be executed. If the initialization file is not available, the log-in attempt will fail. If IINIT was set to OFF by the PN manager (see Section 9.2.1) and the user specifies INIT upon log-in, an explanatory message will be displayed.

9.2.4
Execution of the Initialization File

Upon log-in, the initialization file is attached as local file INITFIL. If the attach fails for any reason, the following message is displayed:

```
PFN=INITFILEFORPNnnnnnINTERACTIVE
CANNOT ATTACH INITIALIZATION FILE.
```

If the initialization file is successfully attached, the system checks to see whether it contains a program or a control statement sequence.

If the file contains images of control statements, the system executes 'EXEC,INITFIL.'

If the file INITFIL contains a program, and the REQUIRED option is set, the following commands are executed by the system:

```
INITFIL.
EXIT,C,S.
LOGOUT.
```

This means that the user is forced to log out immediately after the initialization file program has executed. If the PN manager wishes to override this restriction, the program on INITFIL must issue an EXECM call with the control statement images needed to continue the job (see the *SCOPE/HUSTLER Reference Manual*, Section 8.5.15).

If the file begins with an End-of-Section (EOS) or End-of-Partition (EOP) it will not be used; this will cause the log-in attempt to fail if the REQUIRED option has been set.

9.2.5
Example of Auto-Exec Use

As an example of an appropriate use of the auto-exec procedure, suppose a mathematics instructor has a series of ten self-teaching lessons stored on the permanent file MATHPROBLEMS. He wishes to restrict the use of the class PN to these lessons.

The following portion of an interactive terminal session shows how the instructor could set up this account as desired.

```
OK-SYSTEM GENERAL.
OK-N.
100=ATTACH,MATH,MATHPROBLEMS.
110=MATH.
120=EXIT,S,C.
130=LOGOUT,T.
140==
EON-PROCESSING TEXT
OK-SAVE,ABC,NS.
OK-AUTHORF.

AUTHORF CALLED BY 12345/MATHPROF          ON 9/21/77

CMD? CHANGE IINIT TO LFN=ABC REQUIRED
CMD? END
OK-
```

A sample terminal session by a user on that problem number would look like the following.

```
  15:15:15 01/16/81   MSU-FREND    04.13    SOCKET= 55
[PORT  25]

01/16/81   MSU HUSTLER 2        LSD 50.26   01/13/81   CYBER750

TYPE PASSWORD, PN, AND USER ID.
**********, 7012334,SMITH,S.

SS90135,USER   2  (S  3,P  5)

     MATHPROBLEMS-WHICH LESSON DO YOU WANT?1

     LESSON 1:
     .
     .
     .
     END OF LESSON 1.  DO YOU WISH TO GO ON TO LESSON 2?NO

JOB COST:  $       1.24
```

Appendix A

Character Sets

ASCII Character Set

character	octal	hexadecimal	character	octal	hexadecimal
NUL	000	00	0	060	30
SOH	001	01	1	061	31
STX	002	02	2	062	32
ETX	003	03	3	063	33
EOT	004	04	4	064	34
ENQ	005	05	5	065	35
ACK	006	06	6	066	36
BEL	007	07	7	067	37
BS	010	08	8	070	38
HT	011	09	9	071	39
LF	012	0A	:	072	3A
VT	013	0B	;	073	3B
FF	014	0C	<	074	3C
CR	015	0D	=	075	3D
SO	016	0E	>	076	3E
SI	017	0F	?	077	3F
DLE	020	10	@	100	40
DC1	021	11	A	101	41
DC2	022	12	B	102	42
DC3	023	13	C	103	43
DC4	024	14	D	104	44
NAK	025	15	E	105	45
SYN	026	16	F	106	46
ETB	027	17	G	107	47
CAN	030	18	H	110	48
EM	031	19	I	111	49
SUB	032	1A	J	112	4A
ESC	033	1B	K	113	4B
FS	034	1C	L	114	4C
GS	035	1D	M	115	4D
RS	036	1E	N	116	4E
US	037	1F	O	117	4F
SP	040	20	P	120	50
!	041	21	Q	121	51
"	042	22	R	122	52
#	043	23	S	123	53
$	044	24	T	124	54
%	045	25	U	125	55
&	046	26	V	126	56
'	047	27	W	127	57
(	050	28	X	130	58
)	051	29	Y	131	59
*	052	2A	Z	132	5A
+	053	2B	[	133	5B
,	054	2C	\	134	5C
-	055	2D	]	135	5D
.	056	2E	^	136	5E
/	057	2F	_	137	5F

character	octal	hexadecimal
`	140	60
a	141	61
b	142	62
c	143	63
d	144	64
e	145	65
f	146	66
g	147	67
h	150	68
i	151	69
j	152	6A
k	153	6B
l	154	6C
m	155	6D
n	156	6E
o	157	6F
p	160	70
q	161	71
r	162	72
s	163	73
t	164	74
u	165	75
v	166	76
w	167	77
x	170	78
y	171	79
z	172	7A
{	173	7B
|	174	7C
}	175	7D
~	176	7E
DEL	177	7F

Abbreviations for Control Characters:

NUL	null, or all zeros
SOH	start of heading
STX	start of text
ETX	end of text
EOT	end of transmission
ENQ	enquiry
ACK	acknowledge
BEL	bell
BS	backspace
HT	horizontal tabulation
LF	line feed
VT	vertical tabulation
FF	form feed
CR	carriage return
SO	shift out
SI	shift in
DLE	data link escape
DC1	device control 1
DC2	device control 2
DC3	device control 3
DC4	device control 4
NAK	negative acknowledge
SYN	synchronous idle
ETB	end of transmission block
CAN	cancel
EM	end of medium
SUB	substitute
ESC	escape
FS	file separator
GS	group separator
RS	record separator
US	unit separator
SP	space
DEL	delete

Display Code

Display Code (octal)	Graphic	Display Code (octal)	Graphic
00	none	40	5
01	A	41	6
02	B	42	7
03	C	43	8
04	D	44	9
05	E	45	+
06	F	46	-
07	G	47	*
10	H	50	/
11	I	51	(
12	J	52	)
13	K	53	$
14	L	54	=
15	M	55	blank
16	N	56	,
17	O	57	.
20	P	60	#
21	Q	61	[
22	R	62	]
23	S	63	:
24	T	64	"
25	U	65	_
26	V	66	!
27	W	67	&
30	X	70	'
31	Y	71	?
32	Z	72	<
33	0	73	>
34	1	74	@
35	2	75	\
36	3	76	^
37	4	77	;

Translation from ASCII to APL:

ASCII CODE	CHARACTER	APL TYPE (hexadecimal)	APL BIT (hexadecimal)
20	blank	20	20
21	!	1	1
22	"	4B	4B
23	/	–	–
24	$	7E	7C
25	%	–	–
26	&	–	–
27	'	4B	4B
28	(	3A	2B
29	)	22	2A
2A	*	50	50
2B	+	2D	2D
2C	,	2C	2C
2D	-	5F	3D
2E	.	2E	2E
2F	/	2F	2F
30	0	30	30
31	1	31	31
32	2	32	32
33	3	33	33
34	4	34	34
35	5	35	35
36	6	36	36
37	7	37	37
38	8	38	38
39	9	39	39
3A	:	3E	3E
3B	;	3C	3C
3C	<	23	23
3D	=	25	25
3E	>	26	27
3F	?	51	51
40	@	–	–
41	A	61	61
42	B	62	62
43	C	63	63
44	D	64	64
45	E	65	65
46	F	66	66
47	G	67	67
48	H	68	68
49	I	69	69
4A	J	6A	6A
4B	K	6B	6B
4C	L	6C	6C
4D	M	6D	6D
4E	N	6E	6E
4F	O	6F	6F
50	P	70	70
51	Q	71	71
52	T	72	72
53	S	73	73
54	T	74	74

ASCII CODE	CHARACTER	APL TYPE (hexadecimal)	APL BIT (hexadecimal)
55	U	75	75
56	V	76	76
57	W	77	77
58	X	78	78
59	Y	79	79
5A	Z	7A	7A
5B	[	3B	3B
5C	\	3F	3F
5D	]	27	3A
5E	^	29	5F
5F	_	46	46
60	`	-	-
61	a	61	61
62	b	62	62
63	c	63	63
64	d	64	64
65	e	65	65
66	f	66	66
67	g	67	67
68	h	68	68
69	i	69	69
6A	j	6A	6A
6B	k	6B	6B
6C	l	6C	6C
6D	m	6D	6D
6E	n	6E	6E
6F	o	6F	6F
70	p	70	70
71	q	71	71
72	r	72	72
73	s	73	73
74	t	74	74
75	u	75	75
76	v	76	76
77	w	77	77
78	x	78	78
79	y	79	79
7A	z	7A	7A
7B	{	7B	5D
7C	|	4D	4D
7D	}	7D	7D
7E	~	54	54
7F	DEL	7F	7F

Codes 00 through 1F are sent without translation.

[1] The exclamation is sent to APL terminals as an "overstruck" character, using the three character sequence: slash ('), <backspace>, period (.).

Appendix B
Terminal Types

TERMINAL NAME	TERMINAL IDENTIFIER	RM	CR	LF	HT	VT	FF
Anderson-Jacobson 630	AJ630	140	4	0	1	1	1
Anderson-Jacobson 830	AJ830	130	0	0	0	0	0
Computer Devices Incorporated	CDI	79	4	0	0	0	0
Control Data 713	CDC713	80	0	0	0	0	0
DECwriter II LA36	DEC	132	4	0	0	0	0
DECwriter II LA36 w/APL option	DECAPL	132	4	0	0	0	0
General Electric Terminet 30	GE30	132	0	2	2	2	2
General Electric Terminet 300	GE300	118	0	2	3	3	3
Hazeltine 2000	H2000	79	0	0	0	0	0
Minicomputer	MINI	0	0	0	0	0	0
Ontel	Ontel	80	0	0	0	0	0
Research, Inc. Teleray 3300 Series	Telray	80	0	0	0	0	0
Research, Inc. Teleray 3900 Series	Telray	80	0	0	0	0	0
Research, Inc. Teleray w/APL option	TELAPL	80	0	0	0	0	0
Tektronix 4002	T4002	83	0	0	0	0	0
Tektronix 4010	T4010	73	0	0	0	0	0
Tektronix 4014	T4014	73	0	0	0	0	0
Tektronix 4015	T4015	73	0	0	0	0	0
Tektronix 4023	T4023	80	0	0	0	0	0
Tektronix 4051	T4051	73	0	0	0	0	0
Teletype ASR 33	TTY33	72	2	0	2	2	2
Teletype ASR 35	TTY35	72	2	0	10	10	10
Teletype ASR 37	TTY37	72	2	0	2	2	2
Teletype ASR 38	TTY38	132	2	0	10	10	10
Texas Instruments 700 Series	TI700	79	4	0	0	0	0
General CRT	CRT	80	0	0	0	0	0
General Teletype-like device	TTY	72	2	0	2	2	2

Appendix C
Transmission of 8-Bit Binary Data

This appendix discusses the steps necessary to input and output 8-bit binary data.

BI (Binary) Files

A BI file is a type of connected file on the mainframe which provides for a full 8-bit character set. Data is packed in bits 0-7 of a 12-bit byte, 5 bytes per central memory word. End-of-line is indicated by a byte containing 4000_8 in bits 0-11 of a word. The last word of every line is always padded with 4000_8. See Section 5.1.4 for further discussion.

Front-End Commands

The following Front-End commands are always necessary for binary transmission.

%PARITY,NONE — This command turns off all parity checking allowing the Front-End to treat all 8 bits of each character as data.

%READER,ON — If READER is ON, the Front-End sends a DC3 to tell the transmitter to stop transmission and issues a DC1 to start transmission. This is dependent on the condition of the input buffer. This prevents data from being sent faster than the Front-end and the mainframe can accept it. (The TPREAD command automatically issues a '%READER,ON' command.)

%BINARY — This command disables all special characters. All control characters are transmitted and interpreted as data. Since there is no end-of-line character, data is broken into lines of length INLEN (default=240) before being sent to the mainframe. If the mainframe is reading the data into a BI file, and INLEN is a multiple of 5, no end-of-line indication will be returned (i.e. no 4000_8 byte in the BI file), allowing transmission of a continuous stream of data. After the '%BINARY' command is entered, no further commands will be recognized. Depressing the BREAK key will take the terminal out of binary input mode.

The %BINARY command does not affect whether the parity bit is transmitted. That determination is based on the file type. The parity bit is sent to the user file only if the file is a BI file. Conversely, use of BI files does not disable control characters; only %BINARY can do that.

Note: The %BINARY command does not automatically imply that data is being read on the mainframe into a BI type file: similarly reading from a BI file on the mainframe does not automatically cause the Front-End to behave as if the %BINARY command has been entered.

Reading Binary Data

Data can be read in two fashions: 1) indirectly from a disk file, or 2) directly from a connected file.

1) Indirectly from a disk file

This is the most efficient way to read binary data. The mainframe reads data from the Front-End and writes it to a disk file. This file can then be processed by a user job. This allows the mainframe to process data at the maximum permissible rate.

Command sequence:

```
%PARITY,NONE
TPREAD,lfn,BI.
%BINARY
(read the data)
(depress the BREAK key)
```

2) Directly from a connected file

Your program may read directly from a connected file of type BI. This is slower than reading binary data from disk, because the job must swap in to read each line.

Command sequence:

```
CONNECT,lfn=BI.
%PARITY,NONE
(start user job)
%BINARY
(read data)
(depress BREAK key)
```

Writing Binary Data

Binary data can be written in two ways: 1) indirectly from a disk file, or 2) directly to a connected file.

1) Indirectly from a disk file

This is the most efficient way to write binary data. The mainframe creates disk file TTYTTY, which is automatically written to the Front-End at the completion of the current job step.

In the following examples, program ABC writes data to a disk file.

a) writing to disk file FFF:

```
ABC.                creates file FFF.
WRITEPT,FFF,BI.     copies file FFF to TTYTTY, giving it type BI.
```

b) writing directly to file TTYTTY:

```
SETCODE(TTYTTY=BI)  sets TTYTTY to type BI.
ABC.                creates TTYTTY.
```

2) Directly to a connected file

This is less efficient than writing binary data indirectly from a disk file because the job swaps out every 20 lines. To write directly to a connected file, you need only connect the file as type BI.

Appendix D

Error Messages

Front-End Command Errors

Error Code

UNRECOGNIZED COMMAND (1)
Command was not a legal Front—End Command.

INVALID KEYWORD (2)
Keyword is not valid.

ALTER DOES NOT SUPPORT THIS CHARACTER (3)
In the ALTER,xx=yy command, xx is not a valid character or abbreviation supported by ALTER.

ILLEGAL ALTER FUNCTION (4)
In ALTER,xx=yy, yy is an invalid function name.

KEYWORD MUST BE ON OR OFF (5)
Keyword must be a mnemonic value defined as a parameter for this command, i.e. one of the values which appear in capital letters in the command syntax; in this case "ON" or "OFF."

ILLEGAL TERMINAL TYPE (6)
For TERMINAL,type x is not a legal name.

CHARACTER DELAY MUST BE 0 TO 255 (7)
For CDELAY,xx=n, cc must be 0 to 255.

UNAUTHORIZED COMMAND (8)
Command is not allowed from a user terminal.

VALUE MUST BE 0 TO 240 (9)
For RMARGIN,value. and INLEN,value. the value was >240.

YOU HAVE ONLY 1 CONNECTION (10)
Attempt to FLIP when the user has only one connection; there is no alternate connection to flip to.

If the command was entered on the Front-End computer, any of the preceding error messages is followed by:

COMMAND=cmdname KEY=keywrd

where:

cmdname = command name
keywrd = last keyword or parameter encountered when the error was detected.

For commands issued on the main computer by the FECMD control statement only the single line error message is sent to you as a dayfile message. An error code is returned for each message as indicated in the preceding table.

OTHER FRONT-END MESSAGES

From %JOBSTAT command

JOBSTAT INVALID FOR NETWORK JOB
SERVICE HAS BEEN INTERRUPTED.

From %NETCNT

INVALID HOST ID
INVALID NETCNT PARAMETER
INVALID NETWORK HOST
You indicated a destination other than UM, or WU.

ALL OUTBOUND MERIT PORTS ARE IN USE

"NETCNT, MS" IS NOT ALLOWED. USE "%LOGIN."
Users at MSU are not allowed to use the Merit Network to access the MS host. The command %LOGIN provides the same service at lower system overhead.

[PORT nn]

issued on every %FLIP, %MSU, %NET, %LOGIN, %NETCNT, %QUIT, and upon initially logging in. If you have two connections and log out on one of them, the PORT number is displayed. In short, nn is the number of the primary port to which you are connected, and the PORT message is issued whenever the primary port number changes.

ALL MISTIC CONNECTIONS ARE CURRENTLY IN USE

No interactive connections are available.

ALL MSU INTERACTIVE PORTS ARE BUSY

No more interactive connections are available to MSU interactive users.

BAD DESTINATION CODE

Indicates a Front-End system problem. Please take output to a consultant.

DUE TO SYSTEM LOAD, NO ADDITIONAL LOGINS ARE PERMITTED

The current system load is too high to allow more users to have multiple connections.

\\\INPUT FULL\\\

Five input lines are now waiting in the input queue for the main computer. Any additional input lines will be discarded.

\\\INPUT LINE DISCARDED\\\

The input buffers are full and the line just entered (the sixth line) has been discarded.

MERIT NETWORK UNAVAILABLE
The Merit Network is not available through the Front—End and the main computer.

MISTIC2 | SERVICE HAS BEEN TERMINATED
MERIT |
Issued when interactive or Merit Network service is deliberately stopped by the operator.

MISTIC2 | SERVICE HAS BEEN INTERRUPTED
MERIT |
Issued when there is a problem on the main computer, and interactive or network service is not communicating with the Front-End.

MISTIC2 | SERVICE HAS BEEN RESTORED
MERIT |
Issued when the main computer resumes communicating with the Front—End.

MISTIC2 SERVICE IS NOT AVAILABLE
No interactive service is available.

NO END OF INPUT CHARACTER DEFINED
The end of line character must be defined at all times. Specify an end of line character.

"SECONDARY CONNECTION NOT TO MSU"
Issued when you attempt to use %MSU to flip to your secondary connection when the secondary connection is either not formed through the network or not to host MS. Use %FLIP or %NET.

"SECONDARY CONNECTION NOT TO NETWORK"
Issued when you attempt to use %NET to flip to your secondary connection when the secondary connection was not formed through the network. Use %FLIP.

THERE ARE NO OPEN LINES INTO THE COMPUTER AT THIS TIME.
TYPE %LOGIN UNTIL YOU GET A CONNECTION.
No more interactive connections are available to MSU interactive users.

YOU ALREADY HAVE 2 CONNECTIONS
No more than two interactive connections are allowed per terminal. This message is issued when you attempt to use %NETCNT or %LOGIN when you already have the maximum number of connections.

General Error Messages

CONTROL CARD ERROR - PLEASE TYPE "DAYFILE."
This general message covers bad control statement format and certain illegal parameters. Typing "DAYFILE." will often produce additional diagnostics. This diagnostic generally results from typographical errors.

CPU ABORT
The running program requested job abort; this is usually preceded by additional diagnostics.

DMP CALL INVALID-FIELD LENGTH HAS BEEN RELEASED

Occurs when DMP is typed following a READY, instead of following an EXIT on the same line as a program execution command.

ERROR MODE n AT ADDRESS = aaaaaa

Where n = 3, 5, 6, or 7 — a combination of mode errors 1, 2, and/or 4; e.g., a Mode 5 error is a Mode 1 error and a Mode 4 error.

ERROR MODE 0 AT aaaaaa

This error is equivalent to a program stop. It may indicate a jump to location 0 or an attempt to execute an illegal instruction.

ERROR MODE 1 AT aaaaaa (ADDRESS OUT OF RANGE)

When aaaaaa = 4aaaaa, usually means unsatisfied external at address aaaaa. Unsatisfied externals are always listed if DAYMSG is PART or ON. If DAYMSG is OFF, type MAP(PART) and do the load again to list UNSATISFIED externals. When aaaaaa is some other address value, it usually means a subscript expression is out of bounds.

ERROR MODE 2 AT aaaaaa (INFINITE FLOATING POINT OPERAND)

The instruction located at address aaaaaa is attempting to perform an arithmetic operation with an infinite operand. Infinite operands are the result of a previous operation, usually division by zero.

ERROR MODE 4 AT aaaaaa (UNINITIALIZED OR INDEFINITE OPERAND)

The instruction located at address aaaaaa is attempting to perform an arithmetic operation with an indefinite operand. Indefinite operands are the result of a previous operation, usually 0/0, or the use of an uninitialized variable.

FIELD LENGTH TOO SMALL

Occurs if a system routine is not given enough memory, the result of AUTORFL not ON, or too small a maximum field length specified in the Authorization File or on an MFL statement.

HUNG IN AUTO—RECALL

Occurs when program is waiting for I/O or other system functions and no system activity remains. Often occurs when mixing COMPASS and FTN I/O for the same FET.

(nnn)?

Type-in contained an invalid character in column nnn. The entire input line is thrown away.

OPERATOR DROP

Occurs when an operator types DROP or RERUN for your job. The job would have been detrimental to the system. This message will not normally appear at an interactive terminal.

PP ABORT

A request for system action (e.g., I/O) was ill-formed or illegal; typing DAYFILE will usually provide further diagnostics.

PP CALL ERROR

Occurs when a system action request is invalid.

PP CALL WITH STATUS COMPLETE
Occurs when PP call is made with recall and the complete bit (bit zero of the parameter word or FET) is not clear. Usually caused by a COMPASS logic error.

PROGRAM STOP AT XXXXXX
Program is executing a program stop (zero word).

RTL VALUE EXCEEDS MAX TIME LIMIT
Occurs if desired RTL value exceeds the user's maximum time limit from the authorization file.

TIME LIMIT
CPU time for this control card exceeded the RTL value; program may be in a loop.

UNAUTHORIZED PROGRAM
Occurs when your access level is not high enough for this particular system routine.

USER ABORT
This occurs when the user has aborted his own job with the abort character.

(WAIT SYSTEM)
There are insufficient resources to execute the user's job at this time. Wait for resources to become available.

YOU HAVE USED UP YOUR FUNDS. YOU HAVE ONE DOLLAR TO SAVE YOUR FILES.
Your first dollar limit has been hit. The system finds an exit statement or your program reprieves. Control is regained, and processing continues. There is a second dollar limit of $1.00 assigned to this continued processing.

YOUR FINAL DOLLAR IS GONE. GOODBYE.
User is logged out.

I/O Error Messages

These messages are output for all I/O errors (whether fatal or non-fatal); they are followed by a suggestion to type "DAYFILE." which will provide the file name and FET address.

BUFFER NOT WITHIN FIELD LENGTH
Limit pointer in FET points out of field length.

BUFFER PARAMETER ERROR IN INDEX
Incorrect FET word 8.

CANNOT CONNECT PERM FILE
Self-explanatory.

CIO CODE NOT DEFINED ON DEVICE
For example, attempting to rewrite on a connected file.

ENTIRE INDEX NOT IN FIELD LENGTH-2
Incorrect FET word 8.

ERROR CONDITION NOT CLEARED LAST REQUEST
Program did not clear error status bits set in FET by last operation.

EVICT NOT ALLOWED ON PERMANENT FILES
EVICT releases space assigned to the file.

EXTEND PERMISSION NOT SET FOR RE-WRITE ON P.F.
Trying to write a permanent file without extend permission.

FET BUFFER POINTERS IN ERROR
Means one or more of the following necessary conditions is not met:

FIRST≤IN<LIMIT FIRST≤OUT<LIMIT

FET OUTSIDE FIELD LENGTH
Self-explanatory.

FET POINTERS INCONSISTENT
FIRST, IN, OUT, LIMIT must all be less than the job field length. IN and OUT must be less than LIMIT and greater than or equal to FIRST.

ILLEGAL DEVICE TYPE SPECIFIED
Device other than 02, 13, or 14 specified in FET.

ILLEGAL FILE NAME
File name must be alphabetic character followed by 0-6 alphanumeric characters, in left-justified, zero filled, display-coded format in FET.

ILLEGAL FUNCTION CODE OR ILLEGAL REQUEST
Bad CIO request.

INDEX ADDRESS NOT SPECIFIED FOR RANDOM FILE
Incorrect FET word eight.

INDEX ADDRESS NOT IN FIELD LENGTH
Incorrect FET word eight.

INVALID CONNECT TYPE
Connect type is not DC, AS, AF, BI or BF.

INVALID I/O CODE FOR OPEN
Open request made with code for which no action is defined.

I/O ATTEMPTED ON INCOMPLETE CYCLE OF PF
You've attempted to read or modify a purged PF, but have not specified read or modify permissions on the ATTACH command.

MODIFY PERMISSION NOT SET FOR RE-WRITE ON P.F.
This occurs if one tries to rewrite a permanent file without MODIFY permission.

MSX - OBSOLETE MESSAGE NUMBER
System error - should not occur.

NO PERMISSION SET FOR OPEN READ ON PERM FILE
One cannot open any permanent file without READ permission.

NO READ PERMISSION ON THIS PERM FILE
One cannot read any permanent file without READ permission.

NO REQUEST ENTRY
Device type in FET greater than 37B.

PERMANENT FILE NOT POSITIONED CORRECTLY FOR WRITE
PF must be at EOI for a write operation.

READ OR SKIPF AFTER WRITE
Read is invalid after a write as a write will release all information beyond the write.

Manager Error Messages

BAD READPT/TPREAD CHARACTER SET
Character set was not OM, AS, AF, or BI.

BAD READPT/TPREAD FILE NAME
File name was not seven characters or less, first character alphabetic, all others alphanumeric.

EON-PROCESSING TEXT
End-of-Numbering: the text editor is merging the numbered text lines onto the EWFILE.

EOT-PROCESSING TAPE
Following TAPE or TAPEC, the text lines are being merged into the EWFILE.

Following READPT or TPREAD, the input is being copied to the specified disk file.

INVALID "N" COMMAND - TYPE "HELP(N)".
N did not conform to the rules.

NO TEXT LINES HAVE BEEN ENTERED
Following a TAPE, TAPEC, READPT, TPREAD or N; no data was entered.

TOO MANY CONTROL STATEMENTS ON ONE LINE
When more than one control statement is typed on a line, the statements are expanded into separate card images, each occupying at least one full CM word. Only 64 words are allowed per line when control statements are expanded.

WARNING-TAPE HAD SOME NON-TEXT LINES
This message appears following a TAPE directive. The paper tape contained several lines which did not begin with a line number - these have been ignored.

YOUR CONNECT TIME EXPIRES IN 5 MINUTES
YOUR CONNECT TIME EXPIRES IN 2 MINUTES
An interactive session has a time limit associated with it which governs the maximum time the system may spend running the program. You will receive a warning message five minutes before your connect time expires. You receive another warning message two minutes before expiration.

EDITOR Error Messages

A MOVE OF A RANGE TO A POINT WITHIN ITSELF IS ILLEGAL
MOVE command only. RESEQ can always be used instead, e.g., MOVE,100-200, to 150.

-BY- INCREMENT WILL NOT ASSIGN UNIQUE LINE NUMBERS
In response to the MOVE, DUP, or RESEQ directive, the line increment specified with the BY parameter would generate duplicate EDITOR line numbers on one or more text lines.

CANNOT BLANK COLUMN 0
In intra-line editing, a null text string, signifying column 0 was entered using the blank option. Since column 0 does not exist, it cannot be blanked.

CANNOT DELETE ENTIRE FILE
To prevent a user from accidentally deleting his entire file, it is illegal to do a DELETE without specifying a line range.

CANNOT MOVE ENTIRE FILE

The entire work file was specified in a MOVE operation. This is essentially an undefined operation.

COLUMN NUMBER MUST BE LESS THAN 140

Issued if c_1,c_2 on a search string contains a column number > 140 (does not apply to c_1,c_2 on OLD).

COLUMN NUMBER MUST BE < LENGTH

Issued when the user specifies a column number greater than the line length set by the LENGTH directive.

COMMAND REQUIRES "TO" OR "AT" PARAMETER

The MOVE, INSERT, and DUP directives all require a TO or AT parameter.

DATA RECORD MISSING FROM WORK FILE

The data record representing the indicated line range is missing from EWFILE. Usually this is caused by adding a work file to an APLIB tape without specifying MF.

DUPLICATE LINE NUMBERS IN AN INSERT OPERATION

Because of the size of the file being inserted, it was not possible to assign a unique line number to each inserted line. The resulting work file should be resequenced to ensure unique line numbers.

EDIT INTERRUPTED

A user abort was done after the EDITOR work file had been partially changed. The EDITOR attempts to ensure that the work file remains correct, but this cannot always be done. Therefore, user abort should be used with caution.

EDITED LINE LONGER THAN 700 CHARACTERS

Occurs after an intra-line edit causes the line being processed to become too large for the storage buffer (>700 characters) (also gives line number of offending line).

EDITOR ABORTED - TYPE "DAYFILE"

EDITOR aborted for some reason that is not the EDITOR'S fault — like operator drop, kill, rerun, file limit, dollar limit, etc. Not given for user abort or time limit.

EDITOR FAILURE-SAVE OUTPUT AND NOTIFY CONSULTANT

A system error has occurred in the EDITOR. The terminal output from the current session should be saved and a consultant should be notified so that the problem may be fixed.

EDITOR FILE CAPACITY EXCEEDED

The maximum capacity of the EDITOR work file has been exceeded and all further information will be discarded. The information currently on the work file is still valid and may be edited, saved, etc.

ERROR IN EWFILE - CANNOT CONVERT

Given if EDITOR finds garbage or inconsistencies during conversion. EWFILE remains unconverted.

ERROR IN INFORMATION ON EDITOR WORK FILE - ATTEMPTING RECOVERY

An error existed in the EDITOR work file. The EDITOR will attempt to recover as much as possible of the current work file. A message will indicate the text line range lost due to the file error.

ERROR IN TTYTTY FILE - MISTIC2 SYSTEM ERROR

Occurs if EDITOR reads garbage from TTYTTY. Most common on READPT. This is a system failure, and should always be reported to the consultants.

EWFILE IS LOCKED

Occurs if EWFLOCK,ON is followed by a command which will change the EWFILE. Commands which are EWFLOCK sensitive are: OLD, SCRATCH, INSERT, MERGE, DUP, MOVE, DELETE, INTRA-LINE EDIT, RESEQ, FOLD, and STRING.

EWFILE WILL BE RETURNED, AND A NEW ONE CREATED - TYPE Y TO CONTINUE

In response to an OLD directive, the current EDITOR work file will be returned when the OLD file is read in. Typing Y causes a SCRATCH to be done on the current work file before the OLD operation is done. Typing anything else causes the OLD operation to be bypassed. EWFILE will not be destroyed if it is permanent.

FILE ALREADY EXISTS

Issued by USE, if lfn_2 already exists.

FILE FOR SAVE, OLD, INSERT, OR MERGE CANNOT BE EWF

The EDITOR work file was specified as the source file for a SAVE, OLD, INSERT, or MERGE command. The source file must always be a standard coded sequential file.

FILE IS NOT AN EDITOR WORK FILE. FILE NAME IS lfn

The current file EWFILE, or the file specified in a USE command, is not an EDITOR work file. If this message does not result from a USE command, a SCRATCH can be done to create a new EDITOR Work File.

FROM, BY, TO, AT MUST BE FOLLOWED BY LINE NUMBER

The FROM, BY, TO, or AT options were specified in an EDITOR directive, but the associated line number for the option was missing.

-FROM- GREATER THAN NEXT TEXT LINE

The FROM value in a MOVE, DUP, INSERT or RESEQ operation was greater than the line number of the text line immediately following the line range specified in the directive.

-FROM- LESS THAN LINE NUMBER OF PRECEDING TEXT LINE

When a FROM value is specified in the RESEQ, MOVE, INSERT, or DUP directives, it must be greater than or equal to the line number of the line at which the indicated operation is to begin.

ILLEGAL DIRECTIVE

The indicated command is not a valid EDITOR directive.

ILLEGAL LINE NUMBER RANGE

Occurs for syntax errors like mn--nn or nn-nn-nn.

ILLEGAL PARAMETER: parameter DIRECTIVE directive

The indicated option was illegal for the specified EDITOR directive.

ILLEGAL STRING EDITING OPTION

The editing option specified in an intra-line editing directive was not =, I, B, or L.

ILLEGAL USE OF NULL SEARCH STRING AND COLUMN RANGE

Occurs when "//(c)" is used as first text string in an intra-line edit. Syntactical error — user should just use the "(c)" as the first text string.

INVALID COLUMN NUMBER
The column number in an EDITOR directive was not of the form c_1,c_2 or c_1-c_2 or (c_1), where c_1 and c_2 are the column numbers.

INVALID FILE NAME
In response to any directive which requires a file name, one or both of the following rules were violated: (1) file name must be 1-7 alphanumeric characters or (2) the first character must be alphabetic.

I/O ERROR, TYPE:DAYFILE. FILE IS lfn
An input-output error occurred when reading or writing the indicated file. By typing DAYFILE, the nature of the error will be displayed. Generally, the error is caused by attempting an I/O operation on a permanent file with improper permission.

LENGTH MUST BE 1-140 CHARS
The line length in the LENGTH directive must be between 1 and 140.

LINE NUMBER INCREMENT RESEQ TO .000001
In an INSERT operation, the line number increment for the inserted lines has been reset to .000001 to ensure unique line numbering.

LINE NUMBER OFFSET REQUIRED AFTER + or \
Issued for command containing + or \ not followed by an integer.

LINES MAY BE OUT OF SEQUENCE - MUST RESEQ
Given for EWFLOCK sensitive command after an aborted RESEQ, or an INSERT or ILEDIT in which duplicate line numbers resulted.

LOCK MUST BE ON, OFF, OR PART
EWFLOCK command syntax error.

LOCK(PART) - Y TO CONTINUE
Occurs if EWFLOCK,PART. is followed by SCRATCH, MOVE or RESEQ.

MARGIN NOT EFFECTIVE IN SYSTEM FORTRAN, BASIC
A left margin was set, using the MARGIN directive, while in SYSTEM,BASIC or SYSTEM,FORTRAN. Although the margin will not be in force while in either of these systems, it will be stored, and will become effective when the editing system is changed to SYSTEM,TEXT or SYSTEM,GENERAL. This is a warning message only.

MAXIMUM LINE NUMBER OF 999999.999999 EXCEEDED
The indicated maximum EDITOR line number was exceeded.

NEED EX, MD PERMISSIONS TO CONVERT - RETURN OR NEWNAME EWFILE

NEED EXTEND, MODIFY PERMISSION ON EWFILE FOR THIS COMMAND directive
Affected commands are: OLD, INSERT, MERGE, DUP, MOVE, ILEDIT, RESEQ, FOLD, STRING, FORMAT.

NEED MODIFY PERMISSION ON EWFILE FOR THIS directive
These commands are: TAB, TABCH, SYSTEM, DELETE, LENGTH, MARGIN, EWFLOCK, SET, and any command where the * (line number) parameter is given.

NO EXTEND PERMISSION ON EWFILE PF
When the permanent file was attached, EXTEND permission was not granted. Return the file and reattach with the necessary passwords.

NO FILE NAME FOR "GO" IN SYSTEM GENERAL
The GO command was issued in system GENERAL or TEXT and no EXEC file had been specified in the SYSTEM directive.

NO INCREMENT SMALL ENOUGH TO ASSIGN UNIQUE LINE NUMBERS
Occurs during MOVE, DUP, INSERT, etc.; whenever EDITOR assigns new line numbers to text lines — user should do a RESEQ.

NO SUCH LINE
An attempt was made to DELETE or edit an EDITOR line which did not exist.

NO SUCH STRING
(This message also gives the name of the non-existent string.)

NO TERMINATING STRING DELIMITER
A text string in an EDITOR directive did not have a terminating "/" delimiter.

NO TEXT LINES TO MOVE OR DUP
There were no text lines found satisfying the conditions for the MOVE or DUP operation.

NO TEXT LINES WITHIN RESEQ RANGE
There were no text lines within the line number range specified by a RESEQ, n_1, n_2...

NO 2ND LINE FOR LINE RANGE
A line number followed by a - was specified in an EDITOR directive, but the second number of the line range was missing.

ONLY 1ST INSERT PERFORMED BECAUSE OF -NR- OPTION
Because NR was specified on an INSERT directive, it is not possible to reposition the insertion file to perform the INSERT after the first such operation.

ONLY ONE LINE RANGE IS ALLOWED FOR "RESEQ"
More than one line range specified on RESEQ directive.

ONLY 7 FORMAT TYPES ALLOWED
Given if a FORMAT command will overflow the EWFILE format table. The number of FORMAT ranges does not do this — it is the number of FORMAT types allowed.

ONLY 7 TABS ALLOWED, REMAINDER WILL BE IGNORED
Occurs after TAB directive if too many tab stops were specified.

ONLY 20 LINE NUMBER RANGES ARE ALLOWED
For any directive in which line number ranges can be entered, only up to ten such ranges can be entered.

RECOVERY INTERRUPTED
Occurs when EDITOR's recovery process — i.e. after a user abort, information error on work file, PP call error, earthquake, flood, sunspots — is interrupted. This informs the user before EDITOR tries to continue recovering.

SAVE OF STRINGS REQUIRES "SO" PARAMETER
Missing SOURCE parameter.

2ND COLUMN LESS THAN 1ST
Issued for any c_1, c_2 where $c_1 > c_2$.

2ND LINE LESS THAN FIRST IN LINE NUMBER RANGE
In a line number range, the ending line number in the indicated range was less than the starting line number.

SCRATCH OR NEWNAME CURRENT EWFILE
A USE directive was issued while the current EDITOR work file still contained valid information. The work file should be disposed of through a SCRATCH directive, or saved through a NEWNAME command, before the USE directive can be employed to use a previously created work file.

STRING COMMAND REQUIRES ONE STRING NAME AND ONE TEXT STRING.
Incorrect syntax on STRING directive.

STRING NAME MUST BE 6 CHARACTERS OR LESS
@ followed by over 7 alphanumerics.

SYSTEM MUST BE FORTRAN, BASIC, GENERAL, TEXT, COMPASS, OR BATCH
An illegal editing system was specified in a SYSTEM directive.

TABS AND MARGIN MUST BE LESS THAN LENGTH
All tab stops and the left margin setting must be less than the line length specified in the LENGTH directive.

TABS MUST BE IN ASCENDING ORDER
Tab stops must always be entered in ascending order.

TEXT STRING MUST BE < 140 CHAR
The text string specified in an EDITOR directive must be less than 140 characters long.

TIME LIMIT
The time limit per control statement (RTL value) was exceeded during an EDITOR operation in which the work file had already been altered.

TOO MANY DIGITS IN NUMBER
A line number was entered in an EDITOR directive which contained more than six digits either before or after the decimal point.

WARNING—ASCII DATA EXPECTED ON FILE. lfn MAY BE DISPLAY-CODE
EDITOR has detected a line in the wrong character set. This error may occur when using the OLD, MERGE or INSERT commands.

WARNING—DISPLAY-CODE DATA EXPECTED ON FILE lfn MAY BE ASCII
EDITOR has detected a line in the wrong character set. This error may occur when using the OLD, MERGE or INSERT commands.

WARNING--NO LINES HAVE BEEN "SAVED"
The SAVE directive was used on an empty EWFILE, or no lines fit the criteria specified by the SAVE directive.

WORK FILE IS EMPTY
Issued by GO if EWFILE contains no text.

Appendix E

HELP Categories

HELP entries are classified by a 2 or 3 character code. This classification scheme is based on that used by the Association for Computing Machinery for classifying computer algorithms. Any entry not fitting any of the described subdivisions is given a numeric subclassification of zero. For example, a routine for trinary input would be classified I0 since it would not fit into any of the five defined subcategories of I.

A special group of categories called system categories consists of two alphabetic characters. These categories are used for system utilities, compilers, etc., that would be difficult to locate in the normal categorization scheme.

SYSTEM CATEGORIES

The following categories are directly related to the SCOPE/HUSTLER Operating System, and would be lost if included in any previous categories. Note that there are no numeric subclassifications in this group.

AC.	Accounting and Cost Analysis	Includes rates, accounting and cost analysis for Computer Laboratory Services.
AF.	Authorization File	Includes Authorization file utilities, job authorization mechanisms, and job parameter alteration control cards.
CA.	Compilers and Assemblers	Includes all compilers and assemblers available on the mainframe.
CC.	Miscellaneous Control Cards	Includes all SCOPE/HUSTLER control cards not found in any other category.
ED.	EDITOR Editing Directives	All editing functions available through EDITOR, the interactive text editor.
FE.	File Editing	
FM.	File Manipulation	Includes copy, file positioning, and manipulation commands.
GM.	Games and Novelties	
LB.	Libraries and Library Utilities	Includes utilities for use with HAL and other library maintenance utilities such as UPDATE.
LD.	Loader and Program Call	Includes all loader related commands.
MN.	Merit Computer Network	Information on items pertaining to the Merit Computer Network.

MS.	Interactive System Directives	All interactive system directives not more appropriate in any other special category.
MT.	Magnetic Tapes and Equipment Assignment	Includes magnetic tape utilities.
NW.	News, Information and Assistance	This includes schedules, system news and other useful information.
PF.	Permanent File Utilities	Includes cataloging, attaching, and audit facilities.
SS.	System Stuff	Information on items of interest to sophisticated programmers, such as new macro calls, error codes, etc.
A.	ARITHMETIC ROUTINES	
A1.	Real Numbers	May include multiple precision, fixed and floating-point operations.
A2.	Complex Numbers	May include multiple precision, fixed and floating-point operations.
A3.	Decimal	BCD single or multiple precision arithmetic operations.
A4.	I/O Routines	I/O Routines designed for use with (e.g.) multiple precision arithmetic packages frequently contain entries for both input and output, hence could legitimately be filed under A (filing under I or J therefore merely loses them for prospective users.) While this type of I/O package is not programmed arithmetic in the true sense of the term, its intimate relationship with A-category routines merits that classification.
B.	ELEMENTARY FUNCTIONS	
B1.	Trigonometric	Also pertains to inverse trigonometric functions.
B2.	Hyperbolic	
B3.	Exponential and Logarithmic	
B4.	Roots and Powers	Refers to roots of quantities, not polynomials.
C.	POLYNOMIALS AND SPECIAL FUNCTIONS	
C1.	Evaluation of Polynomials	
C2.	Roots of Polynomials	

C3. Evaluation of Special Functions

C4. Simultaneous Non-linear Algebraic Equations

C5. Simultaneous Transcendental Equations

D. OPERATIONS ON FUNCTIONS AND SOLUTIONS OF DIFFERENTIAL EQUATIONS

D1. Numerical Integration/ Quadrature

D2. Numerical Solutions of Ordinary Differential Equations

D3. Numerical Solutions of Partial Differential Equations

D4. Numerical Differentiation

E. INTERPOLATION AND APPROXIMATIONS

E1. Table Look-up and Interpolation

E2. Curve Fitting

E3. Smoothing

E4. Minimizing or Maximizing a Function

F. OPERATIONS ON MATRICES, VECTORS AND SIMULTANEOUS LINEAR EQUATIONS

F1. Matrix Operations

F2. Eigenvalues and Eigenvectors

F3. Determinants

F4. Simultaneous Linear Equations

G. STATISTICAL ANALYSIS AND PROBABILITY

G1.	Data Reduction	Refers to the calculation of the more common statistical parameters such as mean, median, standard deviation, etc.
G2.	Correlation and Regression Analysis	Includes curve fitting which is explicitly for statistical purposes.

G3.	Sequential Analysis	
G4.	Analysis of Variance	
G5.	Time Series	
G6.	Monte Carlo (See also H)	Includes random number generators.
G7.	Multivariate Analysis	
G8.	Non-parametric statistics	
G9.	Spatial Analysis	
G10.	Special Programs	

H. OPERATIONS RESEARCH TECHNIQUES, SIMULATION AND MANAGEMENT SCIENCE

H1.	Linear Programming	Finding the best solution from among all solutions of a system of linear inequalities.
H2.	Non-linear Programming	Solving constrained optimization problems except those where the objective function and the constraints are all linear.
H3.	Transportation and Network Codes	Transportation codes utilizing efficient solution algorithms. Network codes to find maximal flow through a system.
H4.	Simulation Modeling	"Simulation Modeling" is intended to encompass model components, general simulation programs, and simulation languages; e.g. GPSS or SIMSCRIPT would be categorized "Simulation Modeling." This category covers the tools and technology of simulation.
H5.	Simulation Models	"Simulation Models" is intended to identify completed models of specific systems, however highly parameterized, e.g., a refinery model or computer job shop model would be categorized "Simulation Models". This category covers the completed products of model-making technology—operating models of particular object systems.
H6.	Critical Path Programs	
H8.	Auxiliary Programs	Special purpose utility programs or subprograms designed especially to service programs in the above categories.
H9.	Combined	Programs performing combinations of the above functions.

I. INPUT

I1.	Binary	Pertains to program input or data input in the binary mode (via card, tape, or disk).
I2.	Octal	Pertains to program input or data input in octal mode (via card or tape).
I3.	Decimal	Pertains to program input and data input in the decimal mode (via card or tape).
I4.	BCD (Hollerith)	Pertains to program input or data input in the BCD or Hollerith mode (via card, tape, or disk).
I8.	Free field	A combination of any of the above types of input structured in a free field form.
I9.	Composite	A combination of any of the above, which is not primarily one of the above, such as a general-purpose input program.

J. OUTPUT

J1.	Binary	Pertains to program output (card, tape, or disk) in the binary mode.
J2.	Octal	Pertains to program output (printer) or data output (card or printer) in the octal mode.
J3.	Decimal	Pertains to program output (card, tape or printer) or data output (card, tape, or printer) in the decimal mode.
J4.	BCD (Hollerith)	Pertains to program output (card, tape, printer, or disk) or data output (card, tape, printer, or disk) in the BCD mode.
J5.	Plotting	Refers to routine for producing plotted output, either via printer or via CRT, or other special plotting devices. Routines for using plotting devices to simulate printing are also included.
J7.	Analog	Refers to routines which output information to a digital-to-analog converter, other than that associated directly with (on- or off-line) plotting equipment, which will carry a J5 classification.
J9.	Composite	A combination of any of the above, which is not primary one of the above, such as a general-purpose output program.

K. INTERNAL INFORMATION TRANSFER

Generally denotes core-to-core, tape-to-tape, disk-to-disk, core-to-tape, and core-to-disk movements.

K1.	External-to-External	Pertains to the transfer of information from any external device to any other external device. This would be tape-to-tape, disk-to-disk, disk-to-tape, etc.
K2.	Internal-to-Internal	Pertains to the transfer of information internally. This is the same as relocation of information.
K3.	Disk	Pertains to disk-to-disk, core-to-core, disk-to-core, tape-to-disk and disk-to-tape relocation.
K4.	Tape	Any tape read/write, editing, duplication or comparing, etc. program.
K5.	Direct Data Devices	Computer-to-computer information transfer, other than via the above categories.

L. EXECUTIVE ROUTINES

L1.	Assembly	
L2.	Compiling	
L3.	Monitoring	
L4.	Preprocessing	
L5.	Disassembly and Derelativizing	
L6.	Relativizing	
L7.	Computer Language to Computer Language Translators	This refers to translation from one artificial language designed for computing and data processing purposes to another such language, e.g. FORTRAN to COBOL. Not to be used for translation of natural languages such as English or Russian. See also CA.

M. DATA HANDLING

M1.	Sorting	Combined sort/merge routines are included here.
M2.	Conversion and/or Scaling	Pertains to any conversion and scaling routine (packed or unpacked, single or multiple precision) such as card image to BCD, BCD to card image, binary to BCD, BCD to binary, fixed to floating, etc. The primary function of programs in this category must be conversion or scaling, not input-output.
M3.	Merging	
M4.	Character Manipulation	
M5.	Searching, Seeking, Locating	To be used for utility search subroutines. Not to be used for applications of retrieving information records by examining contents, which is the province of Code S, Information Retrieval.

M6. Report Generators (See also T5)

M9. Composite

N. DEBUGGING

N1. Tracing; Trapping

N2. Dumping — Core, tape, disk, console printouts on- or off-line.

N3. Memory Verification and Searching

N4. Breakpoint printing

O. SIMULATION OF COMPUTERS AND DATA PROCESSORS: INTERPRETERS

O1. Off-line Equipment — Any program which simulates off-line equipment.

O2. Computers — Pertains to programs which simulate or interpret other computers.

O4. Pseudo-Computers — Simulation of theoretical or pseudo-computers.

O5. Software simulation of one peripheral device on another — Includes such programs as simulating tape on disk, simulating card reader on tape, etc.

O9. Other or composite

P. DIAGNOSTICS

Pertains to any program which checks for malfunctioning of the computer or its components.

Q. SERVICE OR HOUSEKEEPING: PROGRAMMING AIDS

Pertains to any routine of utilitarian nature which performs a service for the programmer such as executing the equivalent of pushing a button on the console, setting a dial, or accumulating a check sum.

Q1. Clear/Reset Programs

Q2. Check Sum Accumulation and Correction

Q3. Rewind, Tape Mark, Load Cards, Load Tape Programs, etc.

Q4. Internal House-keeping: Save, Restore, etc.

Q5. Report Generator Subroutines

Q6. Program Documentation: Flow Charter, Document Standardization, etc.

R. LOGICAL AND SYMBOLIC

Logical functions, logical operations, logical calculuses and algebras, symbol manipulation and manipulation of non-numeric quantities.

R1. Formal Logic

R2. Symbol Manipulation

R3. List and String Processing

S. INFORMATION RETRIEVAL

T. APPLICATIONS AND APPLICATION-ORIENTED PROGRAMS

T1. Physics (including nuclear)

T2. Chemistry

T3. Other Physical Sciences (Geology, Astronomy, etc.)

T4. Engineering

T5. Business Data Processing

T6. Manufacturing (non-data) Processing, and Process Control

T7. Mathematics and Applied Mathematics

T8. Social and Behavioral Sciences and Psychology

T9. Biological Sciences

T10. Regional Sciences (Geography, Urban Planning, etc.)

T11.	Computer-assisted Instruction	(includes completed lessons as well as general utilities for constructing lessons)

U. LINGUISTICS AND LANGUAGES

V. GENERAL-PURPOSE UTILITY SUBROUTINES

V2. Combinatorial Generators, Permutations, Combinations and Subsets

X. DATA REDUCTION

Many laboratory or field tests and experiments automatically record data either at the site or by way of telemetry. Programs in this category will accept such digital data and perform the necessary functions of decommutation, scaling, calibrating, evaluating and test analysis. Some of the programs, especially X4, X5, and X6, might be predominantly of type D, E, F, G, and could possibly be found under those categories. The programs are either for post-processing or they may be on-line operation in real time.

X1.	Reformatting, Decommutation, Error Diagnosis	Program separates the variables and often converts them to computer words or higher level languages variables. Missing or erroneous data are identified. Output consists of ordered data and editing information.
X2.	Editing	Output from category X1 is used. Bad data are cast out. missing values inserted, wrong values corrected. Process is either automatic or by parameter cards, or both. Output is called "clean raw data."
X3.	Calibration	Data are scaled linearly, then calibrated to obtain function values in physical units. Output is called "clean calibrated data."
X4.	Evaluation	All necessary computation on the data is performed to present them in a form suitable for engineering or scientific evaluation.
X5.	Analysis	All computations necessary to analyze the outcome of the test or experiment. Also referred to as Time Series analysis.
X6.	Simulation	Programs which generate artificial data to be used as the theoretical text model or to be used for checkout of programs.

Z. ALL OTHERS

This category contains all routines for which no primary class has yet been designated. Routines which are covered by a primary class but which are not adequately described by a subclass are assigned the applicable primary classification with a subclass designation of zero.

Appendix F

Job Field Lengths for System Commands

This table lists the "normal" field length used to execute each system routine. The user FL is used if AUTORFL(PART) is requested.

Name	FL
APL	42000
BASIC	25000
COBOL	65000
COMPASS*	45000
DDL	54000
FORM	53000
FTN*	46000
FTN 5*	53000
IXGEN	65000
LIBEDIT	50000
MNF	46500
REPORT	42000
SORTMRG	54000
SYMPL	60000
UPDATE*	35000

*The field-length listed is not necessarily the minimum required to load; the routine will automatically request additional memory as needed.

Appendix G

Interactive Command and Directive Summary

The following summary lists, in alphabetical order, all EDITOR directives and all commands unique to the interactive system, followed by a list of Front-End commands and control characters. A similar summary of these commands appears in Appendix J of the SCOPE/HUSTLER Reference Manual. Bear in mind that the format descriptions given here are intended as quick-reference aids rather than formal definitions.

NOTATION

UPPER CASE	must appear as shown
lower case	replace with appropriate values
|	separates alternate forms
{ }	encloses alternate forms
[ ]	encloses optional forms
————	indicates acceptable abbreviation
= = = =	default form or value

COMMON PARAMETERS

c	A column number.
(c_1-c_2)	A column range. The forms (c_1,c_2) and (c_1) are also valid.
lnum	A list of one to twenty line numbers and/or line ranges. Legal line numbers go from 0.000001 to 999999.999999. A line range is indicated by two line numbers separated by a dash, e.g. 99.4-126.3. See Section 3.4.1.
lfn	A local file name.
txt	A character string; either a simple character string where txt is defined by /chars/[(c_1[-c_2])][U][N] or a complex character string where txt is defined by txt_1 conj txt_2, where conj is a conjunction. See Section 3.4.3.

EDITING DIRECTIVES SECTION

BASIC[,lnum][,txt][,AF][,CASE][,CTRL][,UNIT]. 3.10.1
BASICX[,lnum][,txt][,AF][,CASE][,CTRL][,UNIT].

Compiles and executes a BASIC program contained in the EDITOR work file (The two forms are equivalent.)

BATCH[,lnum][,txt][,UNIT][,NOSEQ]. 3.11
Disposes a job for batch processing.

COBOL[,lnum][,txt]. 3.10.2
COBOLX[,lnum][,txt].
COBOLER[,lnum][,txt].

Compiles a COBOL program contained in the work file. The COBOL and COBOLER forms will list compiler diagnostics; the COBOLER and COBOLX forms will load and execute the object file (LGO).

COMP[,lnum][,txt][,UNIT][,NOSEQ]. 3.10.3
COMPX[,lnum][,txt][,UNIT][,NOSEQ].
COMPER[,lnum][,txt][,UNIT][,NOSEQ].

Compiles a COMPASS program contained in the EDITOR work file. The COMP and COMPER forms will list compiler diagnostics; the COMPER and COMPX forms will load and execute the object file (LGO).

DELETE[,lnum][,txt][,VETO][,UNIT][,AF][,CASE][,CTRL][,LIST]. 3.13.3

Deletes the text lines specified by the line numbers indicated by lnum and/or the character string text in the EDITOR work file.

DUP[,$lnum_1$][,txt],AT,$lnum_2$[,BY m][,VETO][,UNIT][,AF][,CASE][,CTRL]. 3.13.2

Duplicates all text lines indicated by $lnum_1$ and/or txt at the locations that directly follow the line numbers indicated by $lnum_2$. At each location the duplicated lines must be inserted between the specified line and the next text line of the work file.

EDSTAT. 3.17

Displays current EDITOR work file attributes: the editing system, first and last line numbers, total number of lines, length, margin, tabs, and tab character,etc.

EWFLOCK,[ON|OFF|PART] 3.18

Protects EWFILE from accidental alteration.

FOLD[,lnum][,txt][,UNIT][,VETO][,LIST][,AF][,CASE][,CTRL]. 3.14.4

Truncates all lines which are longer than the current length setting. If line continuation is provided by the current editing system, the remainder of the line is continued on the next line.

FORMAT,sysname[,lnum][,LENGTH n][,TABc1,...cn][,NOSEQ][,HOLD]. 3.15.1
FORMAT.

Defines a line format for groups of lines within the EDITOR work file. FORMAT alone lists all defined formats.

FTN[,lnum][,txt][,UNIT][,NOSEQ]. 3.10.4
FTNX[,lnum][,txt][,UNIT][,NOSEQ].
FTNER[,lnum][,txt][,UNIT][,NOSEQ].

Compiles a FORTRAN Extended or COMPASS program contained in the EDITOR work file. The FTN and FTNER forms will list compiler or assembler diagnostics; the FTNX and FTNER forms will load and execute the object file (LGO).

GO[,execlfn][,lnum][,txt][,UNIT][,NOSEQ]. 3.12

Save the specified text lines on SETFILE and then executes the commands contained in execlfn. If execlfn is omitted, the file name in the last SYSTEM directive is used or, if such a file was not named, a compilation directive is executed, depending on which editing system is in effect.

INSERT,inlfn,AT lnum[,FROM n][,BY m][,NR][,VETO][,AF][,CASE][,CTRL][,txt]. 3.13.5

Inserts the entire contents of inlfn following each of the text lines specified by lnum. At each location the new lines must be inserted between the specified line and the next line of the work file.

LENGTH,c. 3.6.1

Sets the maximum line length to column c. The new length will not alter the current contents of the work file, but it will affect lines subsequently entered, and it may affect the format of files generated from the work file by SAVE, LIST, or PUNCH.

LIST[,lnum|@[abb]][,txt][,AF][,CASE][,CTRL][,NOSEQ][,FULL][,NFULL][,UNIT][,VETO]. 3.8.1

Lists at the terminal the text lines indicated by lnum, with line numbers unless NOSEQ is specified.

LISTF,listlfn[,lnum|@[abb]][,txt][,AF][,CASE][,CTRL][,NOSEQ][,FULL][,UNIT][,VETO]. 3.8.3

Lists text lines indicated by lnum onto file listlfn, with line numbers unless NOSEQ is specified. Extra blanks are suppressed unless FULL is specified.

MARGIN,c. 3.6.2

Sets the left margin to column c. The new margin affects only text lines which are subsequently entered.

MERGE,mrglfn[,FROM n][,BY m][,NR][,VETO][,AF][,CASE][,CTRL][,txt]. 3.13.6

Merges the contents of mrglfn into a non-empty EDITOR work file according to line number. The line numbers associated with the inserted lines may be generated from starting and increment values, or they may be taken from a fixed column range within each line. The default column range is (1,5) for system BASIC and (length, length+14) for all other editing systems.

MOVE[,lnum$_1$][,txt],TO lnum$_2$[,BY m][,AF][,CASE][,CTRL][,UNIT]. 3.13.1

Duplicates the text lines specified by lnum$_1$ and/or txt at each of the locations specified by lnum$_2$ and then deletes the duplicated lines at their original location. At each location specified by lnum$_2$, the duplicated lines must be inserted between the specified line and the next line of the work file.

N[,n][,m]. 3.7.1

Initiates automatic line numbering to simplify entering text lines into EWFILE.

OLD,inlfn[,FROM n][,BY m][,NR][,VETO][,txt][,AF][,CASE][,CTRL][,UPDATE]. 3.7.3

Enters the contents of inlfn into an empty work file.

READ,inlfn[,NR]. 3.7.2

Enters text lines and executes EDITOR directives contained on inlfn.

RESEQ[,lnum][,FROM n][,BY m][,FORMAT]. 3.13.4

Renumbers text lines without altering their order. The default starting value is 100, and the default increment is 10.

SAVE,outlfn[,lnum|@[abb]][,txt][,AF][,CASE][,CTRL][,NOSEQ][,UNIT][,NR][,UPDATE]. 3: 8.2

Copies text lines from the EDITOR work file to outlfn, converting each text line to a SCOPE unit record. The EDITOR line number will be appended to each line (starting in column length+1) unless NOSEQ is specified.

SCRATCH. 3.9.2

Returns the current EWFILE and creates a new one. The work file attributes (system, length, tabs, etc.) remain the same.

SET,param=\{ON|OFF\}[,param=\{ON|OFF\},...]. 3.19

Alters the default setting for EDITOR work file options.

STRING,@abb[,/chars/]. 3.16.1

Defines abbreviations for character strings.

SYSTEM,sysname[,cmdlfn][,UPDATE]. 3.5

Specifies the editing system which formats text lines entered from the terminal keyboard or entered by the directive READ, TAPE, or the system commands TAPEC.

TAB[,c$_1$,c$_2$,...,c$_7$]. 3.6.3

Specifies the column numbers of up to seven tab stops. Any previous tab stops are cleared.

TABCH[,char]. 3.6.4

Defines a tab character, which when encountered in a text line, causes the remainder of the line to be shifted over to the next defined tab stop. The tab character is effective only on text lines entered from the terminal keyboard, from a file processed by READ, or from a paper tape processed by TAPE or TAPEC.

USE,lfn_1[,lfn_2]. 3.9.3

Changes the file name lfn_1 to EWFILE and initializes the file for EDITOR processing. If the current EWFILE is non-empty, it will be renamed lfn_2.

INTRA-LINE EDITING SECTION

The intra-line editing directive is broken down into four cases here. In each case the set of text lines to be edited is defined by a combination of 'txt_1', 'txt_3', and 'lnum' or '@abb'. If none of these parameters is specified, the operation will be performed on every text line in the work file.

txt_1=txt_2[,lnum|@[abb]][,txt_3][,AF][,CASE][,CTRL][,ALL][,LIST][,HOLD][,VETO][,UNIT]. 3.14

Replaces occurrences of txt_2 with the string txt_2.

txt_1Itxt_2[,lnum|@[abb]] [txt_3][,AF][,CASE][,CTRL][,ALL][,LIST][,HOLD][,VETO][,UNIT]. 3.14

Inserts the string txt_2 after occurrences of the string txt_1.

txt_1B//[,lnum|@[abb]][,txt_2][,AF][,CASE][,CTRL][,ALL][,LIST][,HOLD][,VETO][,UNIT]. 3.14

Replaces occurrences of the string txt_1 with blanks.

txt_1Ltxt_2[,lnum|@[abb]][,txt_3][,AF][,CASE][,CTRL][,ALL][,LIST][,HOLD][,VETO][,UNIT]. 3.14

Inserts string txt_2 before occurrences of the string txt_1.

EDITOR PARAMETERS

The following is an alphabetical list and general discussion of the parameters and options used with EDITOR directives. Most parameter settings can be altered using the SET directive.

AF/NAF

Specifies AF mode, which allows you to use EDITOR to create, edit and output ASCII Fancy files. It can be used on the following directives: OLD, INSERT, MERGE, SAVE, LISTF, LIST, BASIC, BASICX, DELETE, DUP, FOLD, MOVE and intra-line editing.

Default: NAF.

ALL/NALL

Used with intra-line editing, which involves the processing of text strings. If ALL is used with the directive, all occurrences of the character string within a line will be edited. If NALL is used, only the first occurrence of the string within a line will be edited.

Default: NALL.

Abbreviation: A, NA.

BY m

Specifies a line number increment; m defines the interval between line numbers generated while processing the following directives: DUP, INSERT, MERGE, MOVE, OLD and RESEQ. Legal values for m are discussed under each directive.

c

Represents a column number. (c must be an integer value.) It can be used in two ways:

1. *To specify individual columns;*

   c or $c_1, c_2, \ldots c_i$

   In this form, c represents a single column or a series of columns. It is used in the LENGTH, MARGIN and TAB directives.

2. *To specify a column range:*

   (c_1, c_2)

   This form describes a column range. The directive processes data in all columns from c_1 to c_2. This is used in intra-line editing.

CASE/NCASE

In processing ASCII files, CASE allows you to consider the case of a character string when string matching. This parameter is used in conjunction with the AF parameter. The following directives use CASE: OLD, INSERT, MERGE, SAVE, LISTF, LIST, BASIC, BASICX, DELETE, DUP, FOLD, MOVE and intra-line editing.

Default: NCASE.

Abbreviations: C, NC.

CTRL/NCTRL

Prevents the suppression of control characters on string searches and the output of the SAVE, LIST and LISTF directives.

Default: NCTRL.

FORMAT/NFORMAT

Controls the way in which format boundaries are handled when the file is resequenced. An EDITOR work file can be partitioned into segments governed by different formatting systems or conventions. This is accomplished through the use of the FORMAT directive. When a segmented file is resequenced, the user can opt to retain the existing format boundaries by specifying the FORMAT parameter with the RESEQ directive. Then the file is resequenced segment by segment. If NFORMAT is specified, the entire file will be resequenced without taking segmentation into consideration.

Default: NFORMAT.

Abbreviation: FMT, NFMT.

FROM n, AT n or TO n

Defines a starting line number. It can be used with the following directives: DUP, INSERT, MERGE, MOVE, OLD, RESEQ.

Abbreviation: FR n.

FULL/NFULL

Used when listing the contents of EWFILE. NFULL causes lines to be listed in compressed format (two or more consecutive blanks are reduced to a single blank).

Default: NFULL.

Abbreviation: F, NF.

HOLD/NHOLD

Used with intra-line editing, which involves the processing of text strings. If HOLD is used with the directive, words are left in their original column positions after editing (where a word is defined as any sequence of characters delimited by blanks). HOLD is most useful when editing text that has been entered using tab stops. If NHOLD is used with a directive, no attempt is made to retain spacing.

Default: NHOLD.

Abbreviation: H, NH.

lfn

The lfn parameter and its variations specify a local file used in place of, or in addition to, EWFILE. The following directives use lfn: GO, INSERT, LISTF, MERGE, OLD, READ, SAVE, SYSTEM, USE.

LIST/NLIST

Used with intra-line editing and the DELETE and FOLD directives. If LIST is used with the directive, all edited lines will be listed at the terminal. If NLIST is specified, the editing operation will be performed without listing the lines at the terminal.

Default: NLIST.

Abbreviation: L, NL.

lnum

The parameter lnum can define a set of up to twenty line numbers, or line number ranges. In this context, a line number can be any of the following:

1. *An integer or decimal number.*

2. *A line number range is of the form, n-m, where n and m are both line numbers, and where the value of n is less than that of m. An inclusive line number range, n-m refers to all text lines between, and including, those represented by n and m.*

   Exclusive line ranges may be specified by adding an X suffix to n or m or both n and m. This refers to all text lines between n and m but excluding, those represented by nX or mX. The line number associated with the X suffix is excluded.

3. *One of the following special symbols:*

   - * *representing the line most recently processed by the last EDITOR directive or the last line entered as a text line. (The user must be sure that * refers to the intended line.)*
   - *F *representing the first line of the work file.*
   - *L *representing the last line of the work file.*
   - *A *representing all lines in the work file. This is equivalent to *F-*L.*

4. *A line number followed by a line count, and having the form m+n, where m is a line number and n is a 1-6 digit non-zero integer. This form denotes the text line n lines past the text line indicated by m.*

5. *A line count alone, having the form, +n, where n is a 1-6 digit non-zero integer. This form denotes the text line n lines past that indicated by the previous line number in the lnum parameter (a positive line offset).*

6. *A line number followed by a line count and having the form m\n, where m is the line number and n is a 1-6 digit non-zero integer. This form denotes the text line n lines before the text line m (a negative line offset).*

NR/R

Specifies whether or not the file specified by lfn is to be rewound. R indicates rewind; NR indicates no rewind. The file is rewound before a read operation and both before and after a write operation. The directives INSERT, LISTF, MERGE, OLD, READ and SAVE use this option.

Default: R.

SEQ/NOSEQ

Determines whether or not EDITOR line numbers are to be included with the text. It can be used with LIST, LISTF and SAVE.

Default: SEQ.

Abbreviation: S, NS.

SOURCE/NSOURCE

Determines the form in which an EDITOR work file is retained. When the SOURCE parameter is used with the SAVE directive, the EWFILE is stored in a form suitable to be re-entered in an EWFILE using the READ directive. This means that text is stored with line numbers in column 2, separated from the text by an equal sign.

Default: NSOURCE.

Abbreviation: SO, NSO.

txt

The txt parameter defines a character string, which is used to specify which lines of EWFILE are to be processed by the EDITOR directive in question. The txt parameter has the form:

/chars/[(c_1[,c_2])][U][N]

Compound Character Strings

The txt parameter can be used to define compound text search strings. There are four different types of conjunctions.

1. $txt_a + txt_b$
2. $txt_a \& txt_b$
3. $txt_a \$ txt_b$
4. $txt_a \$ nntxt_b$

These have the following effects on lines to be processed by EDITOR.

1. *All lines in which either txt_a or txt_b is satisfied will be processed.*

2. *All lines in which both txt_a and txt_b are satisfied will be processed.*

3. *All lines in which both txt_a and txt_b are satisfied with txt_b following txt_a by any number of characters will be processed.*

4. *All lines in which both txt_a and txt_b are satisfied with txt_b following txt_a by exactly nn characters will be processed.*

A compound character string can be used in any statement that allows the txt parameter, except for the txt_1 and txt_2 parameters in the intra-line editing directives.

UNIT/NUNIT

Used in conjunction with the txt parameter. It specifies whether or not the text search string must appear as a unit within the text line. The UNIT parameter affects all text strings in the directive. It can be used with the following directives: DELETE, DUP, LIST, LISTF, MOVE, SAVE, SET and intra-line editing.

Default: NUNIT.

Abbreviation: U, NU.

UPDATE/NUPDATE

Allows the user to use the CDC utility, UPDATE, with EDITOR. It can be used with the following directives: OLD, SAVE, SET and SYSTEM.

Default: NUPDATE.

Abbreviation: UP, NUP.

VETO/NVETO

Allows the user to examine each line processed by an EDITOR directive before the operation is performed on the line. The user can decide what action to take by typing an action code.

Default: NVETO.

Abbreviation: V, NV.

Code	Action
Y (YES)	*perform the operation on this line. VETO continues.*
N (NO)	*do not perform the operation on this line. VETO continues.*
A (ACCEPT)	*perform the operation on this line, and then terminate the operation.*
S (STOP)	*stop the operation (without editing the current line).*
C (CONTINUE)	*perform the operation on this line and continue processing remaining lines without VETO or LIST.*
L (LIST)	*perform the operation on this line and continue processing remaining lines without VETO but list all processed lines in full.*
K (KILL)	*stop the operation (without editing the current line) then exit EDITOR via a CPU abort.*

Yn — *perform the operation without VETO on the next n lines (including the current line), where n is a 1-5 digit non-zero integer line count. VETO is re-initiated after n lines have been processed.*

Nn — *do not perform the operation on the next n lines (including the current line), where n is a 1-5 digit non-zero integer line count. VETO is re-initiated after the n lines have been processed.*

Ln — *perform the operation without VETO on the next n lines (including the current line), where n is a 1-5 digit non-zero integer line count. All lines following the current line that are affected by the operation will be listed in full. VETO is re-initiated after n lines have been processed.*

=txt — *replace the line with the character string txt. The user may use VETO on this line before proceeding.*

@abb

A user-defined abbreviation for a character string, where abb is the abbreviation. The directive STRING is used to define abbreviations. If @ appears alone, it represents all currently defined abbreviations. Whenever @ appears in a directive, the directive edits all strings that have abbreviations. Five EDITOR directives allow strings to be referenced by their abbreviations for editing purposes. These are DELETE, LIST, LISTF, SAVE and intra-line editing.

INTERACTIVE COMMANDS — SECTION

ASSETS. 2.10.3

Displays user limits and current usage of certain system resources, including field length, CPU time, connect time, and files. Also prints the current status of the AUTHORF, REDUCE, MAP, PROMPT, LOCK and DAYMSG flags.

CONNECT,lfn$_1$[=cc][,...[,lfn$_n$[=cc]]. 5.2

Connects local file lfn. Subsequent read requests on lfn will cause program execution to pause until input is received from the terminal keyboard. Subsequent write requests on lfn will display the information at the terminal rather than copy it to disk.

DAYFILE[,F],fromline,toline[,F]. **6.2.1**

Displays dayfile messages.

DAYMSG,{ON|PART|OFF}. 2.10.4

Enables the user to suppress all or part of the dayfile messages received at the terminal.

DEBUG[,ON|,OFF]. **6.3.5**

Enables the Cyber Interactive Debug facility, placing object code in the debug mode, and loads various CID control MODULES with every relocatable load.

DISCONT,lfn. *5.4*

Disconnects lfn. Subsequent write or read requests on lfn will transfer information to or from disk storage rather than the user's terminal.

DMP[[,fwa],lwa]. 6.3.2

Produces a standard dump.

ERRS[I=inlfn][,O=outlfn][,ALL][,F][,NA][,NI][,NS][,PG=N][,S]. 6.1.1

Scans inlfn for ALGOL, COBOL, COMPASS, FTN, SPSS or UPDATE diagnostics.

EXIT[,S][,C|,U]. 6.5.1

Precedes a group of control statements to be executed in the event of a fatal job error.

FILES. 2.7.2

Displays the names of all local user files and flags the names of permanent and connected files.

LISTAPE. 2.10.5

Lists the visual reel names of all tapes that have been requested but not yet mounted and assigned.

LISTTY,I=listlfn[,NR][,B]{[Ccol-wid]|[,Wwid][,Ccol]}{[,startline-lastline][,Sstartline]
[,Llastline][,O=outlfn]. 2.7.2

Lists selected lines of local file listlfn at the terminal. The user can list a column range by specifying the starting column and ending column.

LOCK,{ON|PART|OFF}. 2.5.3

Enables the user to inhibit the receipt of messages from the operator or other terminals.

LOGOUT[,T]. 1.2

Initiates end-of-session procedures. The T parameter suppresses the display of detailed accounting information on the interactive session and returns all files.

MAP[,ON|,OFF|,PART|,FULL]. 6.4.1

Defines the load map in effect for the remainder of the job. Interactive default is OFF, batch default is PART.

MESSAGE 2.5.4

Sends a message to the operator.

MODE,n. 6.5.2

Selects exit conditions for arithmetic mode errors.

OK. 2.4

Substitutes the "OK-" message for the "READY hh.mm.ss." message.

PAGE[,optional parameters]. 2.6.3

List the contents of a coded file at a terminal, page by page.

PROMPT[,ON|,OFF]. 5.5

Requests the interactive system to print an asterisk as a prompt character whenever an executing program is waiting for terminal input.

READPT,inlfn[,NR][,cc]. 2.9.3

Copies the contents of a paper tape to inlfn.

READY. 2.4

Substitutes the "READY hh.mm.ss." message for the "OK-" message.

RTL,nnn. 2.6.1

Specifies the additional time added to the cumulative CPU time limit for each command executed.

SAVEDMP[,ON|,OFF]. 6.3.3

Produces a full dump on TTYDMP whenever there is an abnormal termination of a user program.

SEND[,id]. 2.5.2

Sends a message to another interactive terminal.

SETCODE(lfn$_1$=cc[,lfn$_2$=...]). 5.3

Sets the character code associated with a local file.

SITUATE. 2.5.1

Lists the IDs of all users currently logged in on the interactive system.

TAPE. 2.9.1

Enters numbered text lines from paper tape.

TAPEC. 2.9.2

Enters numbered text lines and executes EDITOR directives contained on paper tape.

TPREAD,inlfn[,NR][,cc]. 2.9.3

Copies the contents of paper tape to inlfn. Identical to READPT,except that TPREAD can automatically start and stop the tape reader.

TRAP[,I=inlfn][,O=outlfn]. 6.4.2

Causes the execution time debugging routine TRAPPER to be loaded with the user's program.

WRITEPT,inlfn[,cc][,NR]. 2.9.4

Copies the contents of a local file to paper tape.

FRONT-END COMMANDS

SECTION

%ALTCHAR,{charset|OFF|NONE}[,AUTO] 8.2.4

Specifies a terminal as being in an alternate character set.

%ALTER,char=function{,char=function,...} 8.6.2
%ALTER,RESET
%ALTER,LIST

Assigns different functional meaning to control characters. Form 2 resets all control characters to their original function. Form 3 lists all control characters and their function.

%BINARY,{ON|OFF} 8.3.14

Disables all special characters, allowing the transmission of binary data to the main computer system.

%CCTL,{ON|OFF} 8.3.12

Controls the processing of carriage control characters during terminal output.

%CDELAY[,CR=n_1][,LF=n_2][,HT=n_3][,VT=n_4][,FF=n_5] 8.2.3

Sets the number of nulls sent after the transmission of a terminal control function.

%CONSTAT 8.4.2

Displays the user's socket and port numbers.

%DEQ[,n][,LIST] 8.3.4

Deletes input lines which are waiting in the front-end input queue on a last-in/first-out basis.

%ECHO,{ON|OFF} 8.3.4
%ECHO,BKSP=char.

Controls the echo printing of input. (OFF is the default for 110 baud terminals; ON is the default for terminals faster than 110 baud.) You can use the second form of the ECHO command to redefine the character echo printed for BKSP.

%FECC,x 8.6.1

Redefines the Front-End control character.

%FESTAT 8.4.5

Displays the number of interactive jobs currently on the system.

%FLIP 8.5.3

Exchanges the primary and secondary connections when the user has multiple jobs.

%INLEN,n 8.3.9

Sets the length of input lines to n. The default is 240 characters.

%JOBSTAT 8.4.1

Displays the current status of the user's job on the main computer.

%LOGIN 8.5.1

Establishes another connection between the user's terminal and the main computer system.

%LOGINMSG 8.4.6

Displays the current log-in message at the terminal at any time.

%MIX,{ON|OFF} 8.5.4

Controls the transmission of output from multiple jobs to the terminal. The user can receive intermixed output or the output of one job at a time.

%MSU 8.5.3

Exchanges the user's primary and secondary connections. The user's MSU connection becomes primary and the Merit Network connection secondary.

%NET 8.5.3

Exchanges the user's primary and secondary connections. The user's Merit Network connection becomes primary and the MSU connection secondary.

%NETCNT,{UM|WU} 8.5.2

Requests a Merit Computer Network connection to the specified site.

%PARITY,{EVEN|ODD|NONE} 8.2.3

Sets the criterion for the parity checking process.

%QUIT 8.5.6

Disconnects the user's primary interactive job.

%READER,{ON|OFF} 8.3.13

Automatically starts and stops the input of data; used in conjunction with paper tape and other auxiliary input devices.

%RMARGIN,cwidth 8.3.8

Sets maximum output line length in characters; characters beyond cwidth are folded.

%[SHOW|NPC[,{OFF|PART|ON}][,{MNEM|CTRL|char}] 8.3.11

Prints graphic representations of non-printing characters at the terminal.

%TERMINAL,type 8.2.1

Sets the appropriate default values for the attributes of the terminal in use.

%TERMSTAT 8.4.4

Displays current terminal attributes.

%TIME 8.4.3

Displays the current time and date as maintained by the Front-End.

CONTROL CHARACTERS

Character	Function Code	Default	ASCII Code	Section
abort	ABORT	ESC	1B	8.1.1
character delete	BKSP	BS,CTRLH	08	8.1.3
discard output line	TERMOUT	SUB,CTRLZ	1A	8.3.2
echo line	LECHO	ACK,CTRLF	06	8.3.7
end-of-line	EOL	CR,CTRLM	0D	8.1.2
halt output	HALTOUT	DC3,CTRLS	13	8.3.1
ignored character	IGNORE	DEL	7F	8.6.2
line continue	CONT	ETB,CTRLW	17	8.3.10
line delete	CANCEL	CAN,CTRLX	18	8.1.4
literal next	LITERAL	DLE,CTRLP	10	8.1.5
retrieve input	RETIN	NAK,CTRLU	15	8.3.3
start output	STARTOUT	DC1,CTRLQ	11	8.3.1
stop output	STOPOUT	DC4,CTRLT	14	8.3.1
switch echoback	ECHO	SYN,CTRLV	16	8.3.6

APPENDIX H

SAMPLE INTERACTIVE SESSIONS

In this appendix, we present sample interactive sessions. The terminal output from each is annotated to point out some of the subtleties of interactive programming. Unless specified otherwise, the numbers in parentheses are section references to the corresponding procedure in the Interactive System User's Guide.

Sample Session 1 is an example of:

1. creation of a FORTRAN program,
2. correcting source program errors with interactive editing facilities,
3. interactive execution of the program, and
4. the use of CATALOG, LIST, and DISPOSE.

Sample Session 2:

Reserved for a future debugging session

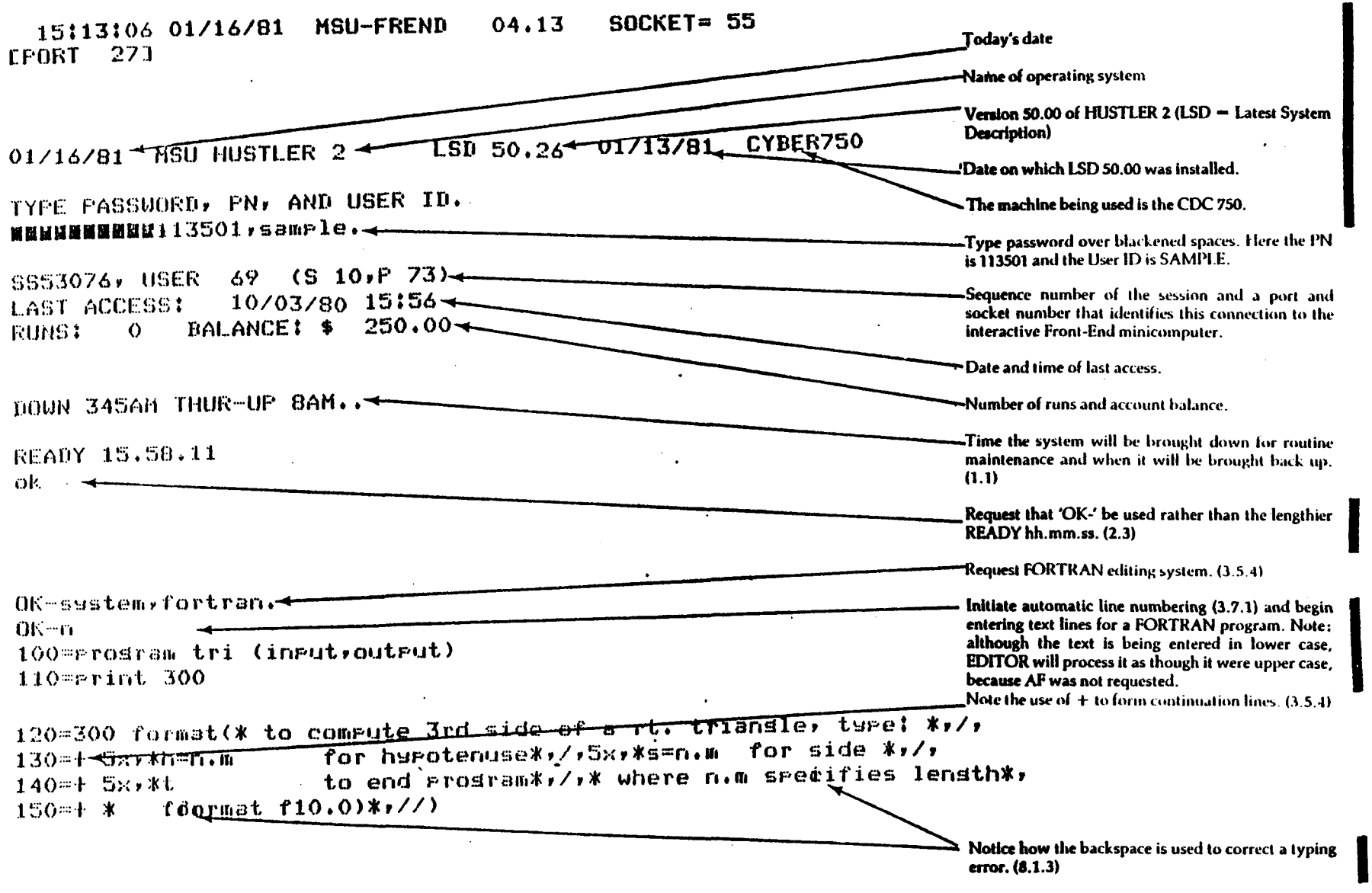
Sample Session 3 displays:

1. two forms of PFLIST,
2. creating and excuting a PFLOAD batch job interactively, and
3. the HAL command, STATUS.

Sample Session 4 displays:

1. a sample SPSS job.

Sample Session 1



```
160=x=-1
170=read 100,11,s1
180=read 100,12,s2
190=if (11.eq.1hh) x=sqrt(s1**2-s2**2)
200=if (12.eq.1hh) x=sqrt(s2**2-s1**2)
210=if (x.lt.0)    x=sqrt(s1**2+s2**2)
220=if (x
=185 if (11.eq.1ht).or.(12.eq.1ht) call exit
220=print 200,x
230=go to 1
240=100 format(a1,1x,f10.0)
250=200 format(e15.5)
260=end
270==ftn4

  COMPILING TRI
         2  FORTRAN ERRORS IN  TRI
       .100 CP SECONDS COMPILATION TIME
 CPU ABORT
 0 INFO ERRORS IN TRI
 FE ILLEGAL SYNTAX AFTER INITIAL KEYWORD OR NAME.
   185
 UNDEFINED STATEMENT NUMBERS: 1


OK-160 1 x=-1
list,185
185= IF (L1.EQ.1HT).OR.(L2.EQ.1HT) CALL EXIT
OK-/)/i/)/,185,veto
185=       IF (L1.EQ.1HT)).OR.(L2.EQ.1HT) CALL EXIT
?n
OK-/) call/=/)) call/,185,v
185=       IF (L1.EQ.1HT).OR.(L2.EQ.1HT)) CALL EXIT
?y
OK-//(/i/(/,185,veto
185=       IF ((L1.EQ.1HT).OR.(L2.EQ.1HT)) CALL EXIT
?y
OK-prompt.
```

Cancel input line with a CTRL-X. (8.1.4)

Prefix type-in with an equal sign to reject automatic line number 220 and insert EDITOR line number 185. (3.7.1)

Prefix type-in with an equal sign in order to issue FTN 4 compilation directive. (3.10.4) (Auto-line numbering terminates).

Oops!

Statement is lacking proper FORTRAN Syntax.

FORTRAN statement number 1 was not defined.

Fix undefined line number by replacing line 160. (3.2).

List line 185 in order to check it. (3.8.1).

FORTRAN requires the entire logical operation to be enclosed in parentheses.

Need to insert a '(' between (and L and a ')' between T and).

Insert a ')' after a ')' in line 185 (3.14.1) and request VETO processing. (3.4.5).

The ')' was inserted after the first occurrence of a ')' in line 185. Reject the edit.

Try again, using a replacement operation in order to specify the second occurrence of ')'. Note the use of 'v' to specify VETO processing. (3.14.1). This time it's o.k. Yes, perform the edit.

Now add the opening parenthesis by inserting a '(' after the first occurrence of '('.
Look's good. Yes, perform the edit.

Turn on automatic prompting feature in preparation for the execution of TRI. (5.5)

```
OK-ftner
 COMPILING TRI
       .234 CP SECONDS COMPILATION TIME
 EXEC BEGUN.16.24.42.
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
       .028 CP SECONDS EXECUTION TIME
OK-catalog,ewfile,triewfile,id=sample,rd=a,md=b,cn=c,ex=d.
 CATALOG,EWFILE,TRIEWFILE,ID=SAMPLE,RD=*---*,MD=*---*,CN=*---*,EX=*---*.
OK-files.
  SETFILE    C*OUTPUT     C*ZZZZOT       TTYTTY
C*ZZZZZIN    P*EWFILE       LGO         C*INPUT
OK-move,120-150,td,252
RESEQ BY 2
LAST NUM 258
OK-list.
100=       PROGRAM TRI (INPUT,OUTPUT)
110=       PRINT 300
160=1      X=-1
170=       READ 100,L1,S1
180=       READ 100,L2,S2
185=       IF ((L1.EQ.1HT).OR.(L2.EQ.1HT)) CALL EXIT
190=       IF (L1.EQ.1HH) X=SQRT(S1**2-S2**2)
200=       IF (L2.EQ.1HH) X=SQRT(S2**2-S1**2)
210=       IF (X.LT.0)    X=SQRT(S1**2+S2**2)
220=       PRINT 200,X
230=       GO TO 1
```

Compile EWFILE, scan for errors, and LGO! (3.10.4)

This is the program's output, which gives instructions for using this routine. Note loop control option, T.

Instructions for format of input——a good practice.

The asterisks are produced by PROMPT to indicate that TRI is ready for the user to type something.

Specify length of hypotenuse as 5.0.
Specify length of other side as 3.0.
Program computes the length of the other side as 4.0.
Specify length of side as 4.0.
Specify length of other side as 3.0.
Length of hypotenuse as 5.0.
Terminate program.

Catalog EWFILE as a permanent file with READ, MODIFY, CONTROL, and EXTEND permissions protected by the passwords A, B, C, and D respectively. (3.9.1).

List names of local files. (2.6.1)

ZZZZIN, OUTPUT, INPUT, and ZZZZOT are connected temporary files. EWFILE is a permanent file. SETFILE, LGO, and TTYTTY are local files.

Move FORMAT statement 300 to the end of the program. (3.13.1). The four lines are assigned line numbers 252, 254, 256, and 258. These changes are automatically made permanent.

List entire contents of EWFILE but don't suppress extra blanks. (3.8.1)

Note that line 185 was inserted between lines 180 and 190.

```
240=100    FORMAT(A1,1X,F10.0)
250=200    FORMAT(E15.5)
252=300    FORMAT(* TO COMPUTE 3RD SIDE OF A RT. TRIANGLE, TYPE: *,/,
254=      + 5X,*H=N.M       FOR HYPOTENUSE*,/,5X,*S=N.M  FOR SIDE *,/,
256=      + 5X,*T           TO END PROGRAM*,/,* WHERE N.M SPECIFIES LENGTH*,
258=      + *    (FORMAT F10.0)*,//)
260=       END
OK-save,out.
OK-dispose,out,pa.
 SUBMITTED UNDER SEQUENCE        SB53076
OK-logout.
PLEASE DISPOSE OF YOUR FILES.
SETFILE?r
LGO     ?d
CP USE       1.606 SEC    VALUE $        .07
PP USE      42.477 SEC    VALUE $        .12
CM USE       1.661 W-H    VALUE $        .45
CT USE        .593 HRS    VALUE $        .89
TOTAL VALUE OF JOB AT RG3     $        1.84
```

Here are the four lines that were moved.

Copy contents of EWFILE to the standard coded file OUT. (3.8.2)

Send OUT to a central site printer. (7.1)

The output should be picked up under this sequence number.

Terminate the session. (1.2.3)

Retain SETFILE (for 2 hours).

Destroy LGO.

Terminate LOGOUT: all other temporary files are destroyed but permanent and common files are saved by the system and retain their permanent or common status.

Central Processor time
Peripheral Processor time
Central Memory usage
Connect Time (actual length of session).

Pages H-6 through H-9 are reserved for a future example debugging run.

Sample Session 3

```
EPORT  213


01/21/81  MSU HUSTLER 2      LSD 50.27  01/15/81  CYBER750

TYPE PASSWORD, PN, AND USER ID.
■■■■■■■■■■

SS22060, USER  12  (S 15,P 21)
LAST ACCESS: S 10/03/80 00:20
RUNS:  80   BALANCE: $ 9161.57


DOWN 345AM WED-UP 8AM.

READY 00.34.25
pflist.
PFLIST.

      LIST OF PERMANENT FILES     TIME  .12.47.33.      01/19/81

PERMANENT FILE NAME   CY   OWNER  CREATE EXPIRE LACC ATT  SIZE  COST
 PN-0111600ANNUALREPORT        FACSERV
                       1           10/20 INFIN 01/24   6    77   .103
 PN-0111600ACRO                FTNTEST
                       1           10/23 INFIN 10/23   0   374   .499
 PN-0111600HELP                1
                       1           11/07 INFIN 02/08  33   770  1.027
 PN-0111600WALKTHROUGH         MAIL
                       1 REDACT    11/17 INFIN 11/17   1    44   .059
 PN-0111600ACRO                FTN5
                       1           10/23 INFIN 10/24   2   407   .543

 TOTAL FILES =     5   TOTAL PRUS =     1672   TOTAL COST/DAY =    2.23
```

Request that a report describing the status of your permanent files be generated. (5.3 in SCOPE/HUSTLER RM)

Normal interactive format using TTYTTY as the default.

- permanent file name
- cycle number
- owner ID
- creation date
- expiration date
- date of last access
- number of attaches
- size of file in PRUs
- cost per day

```
READY 00.36.30
pflist,purged,vrn.
PFLIST,PURGED,VRN.

      LIST OF PERMANENT FILES      TIME  .00.37.17.       02/03/81

PERMANENT FILE NAME    CY  DUMP  VRN1    VRN2    VRN3    VRN4  FLAGS    SIZE
*REDACTFACSERVBACKUP    1 02/10 PFDB01                          1  7      77
*ZORT                   1 02/01 PFS324                          1  7       2

 TOTAL FILES =     2    TOTAL PRUS =       79    TOTAL COST/DAY =     0.00

READY 00.39.01
system,batch.

READY 00.39.15
n
100=*jobcard*,rs2,jc1500,mt1.
110=pfload,pfn=redactfacservbackup,mt=pfdb01,rp=100.
120==
EON-PROCESSING TEXT
(WAIT SYSTEM)

READY 00.40.25
list,nf.
100=*JOBCARD*,RG2,JC1500,MT1.
110=PFLOAD,PFN=REDACTFACSERVBACKUP,MT=PFDB01,RP=100.

READY 00.40.34
go
 SUBMITTED UNDER SEQUENCE       TB22061.

READY 00.40.54
hal,status,tb22061.
 HAL 5.13
TB22061 WAITING TAPE PFDB01   NO RING  _F

READY 00.41.10
%time
  00:41:37 02/14/79
```

Request all your purged files be listed along with the Visual Reel Numbers of the Computer dump tapes they were last dumped on.

Note the differences in the purged file listing.

"'s are purged file flags
DUMP is the date when the file was last dumped to tape.
Visual Reel Number of tape to which the permanent file was dumped.
FLAGS indicating the status of the file dumping operation.

Request BATCH editing system (3.5.2)

Initiate automatic line numbering (3.7.1)

Set-up the job card for the batch job you are going to submit in order to reload some purged files. (3.2.3 in SCOPE/HUSTLER RM)

Reload the purged file REDACTFACSERVBACKUP and retain it for 100 days. (5.4.4 in SCOPE/HUSTLER RM)

Submit your batch job. (3.12)

Check on the job's STATUS. (2.9.2)

Job is waiting for its requested tape to be mounted.

```
hal,status,tb22061.
 HAL 5.13
TB22061 WAITING TAPE PFDB01    NO RING  -P

READY 00.48.10
%time
  00:48:17 01/23/81
hal,status,tb22061.
 HAL 5.13
TB22061 IS WAITING TO PRINT(PR), RG2660, BEHIND      7

READY 00.57.58
attach,x,redactfacservbackup.
 ATTACH,X,REDACTFACSERVBACKUP.

READY 00.58.19
pflist.
PFLIST.

      LIST OF PERMANENT FILES      TIME  .00.58.26.       01/23/81

PERMANENT FILE NAME    CY    OWNER  CREATE EXPIRE LACC ATT  SIZE  COST
 PN-0111600ANNUALREPORT          FACSERV
                        1            10/20 INFIN 01/24   6    77   .103
 PN-0111600ACRO                  FTNTEST
                        1            10/23 INFIN 10/23   0   374   .499
 PN-0111600HELP                  1
                        1            11/07 INFIN 02/08  33   770  1.027
 PN-0111600WALKTHROUGH           MAIL
                        1 REDACT     11/17 INFIN 11/17   1    44   .059
 PN-0111600ACRO                  FTN5
                        1            10/23 INFIN 10/24   2   407   .543
 REDACTFACSERVBACKUP    1*           10/20 05/25 02/14   2    77   .103

 TOTAL FILES =     6    TOTAL PRUS =     1749    TOTAL COST/DAY =     2.33
```

Attach the file in order to verify that it has been reloaded. (5.2.2 in SCOPE/HUSTLER RM)

Do a permanent file listing to double-check.

There is the file you reloaded.

```
FPORT  483


04/16/80 *MSU HUSTLER 2       LSD 49.51  04/13/80  CYBER750

TYPE PASSWORD, FN, AND USER ID.
■■■■■■■■■■■

SS15182, USER  40  (S 65,P 48)
LAST ACCESS: S 04/14/80 16:06
RUNS:  77   BALANCE: $ 9541.65
.
.
.
OK-attach,dat,spssdata.
ATTACH,DAT,SPSSDATA.
OK-system,batch.tab,16.tabch,;.
OK-n,100.
100=variable list;ques1 to ques9
110=input format;fixed(t38,9f2.0)
120=n of cases
130=missing vaulues;all(blank)
140=condescriptive;ques1 to ques9
150=statistics;all
160=read input data
170=save file
180=finish
190=
FOR-PROCESSING TEXT
```

Attaches to your job the permanent file containing data for your SPSS program (SPSSDATA, in this example), and assigns it the local file name "DAT" for use in the job. (Chapter 2 *SCOPE/HUSTLER Reference Manual*)

System echo of the ATTACH control statement indicates that your data file has been attached to the job.

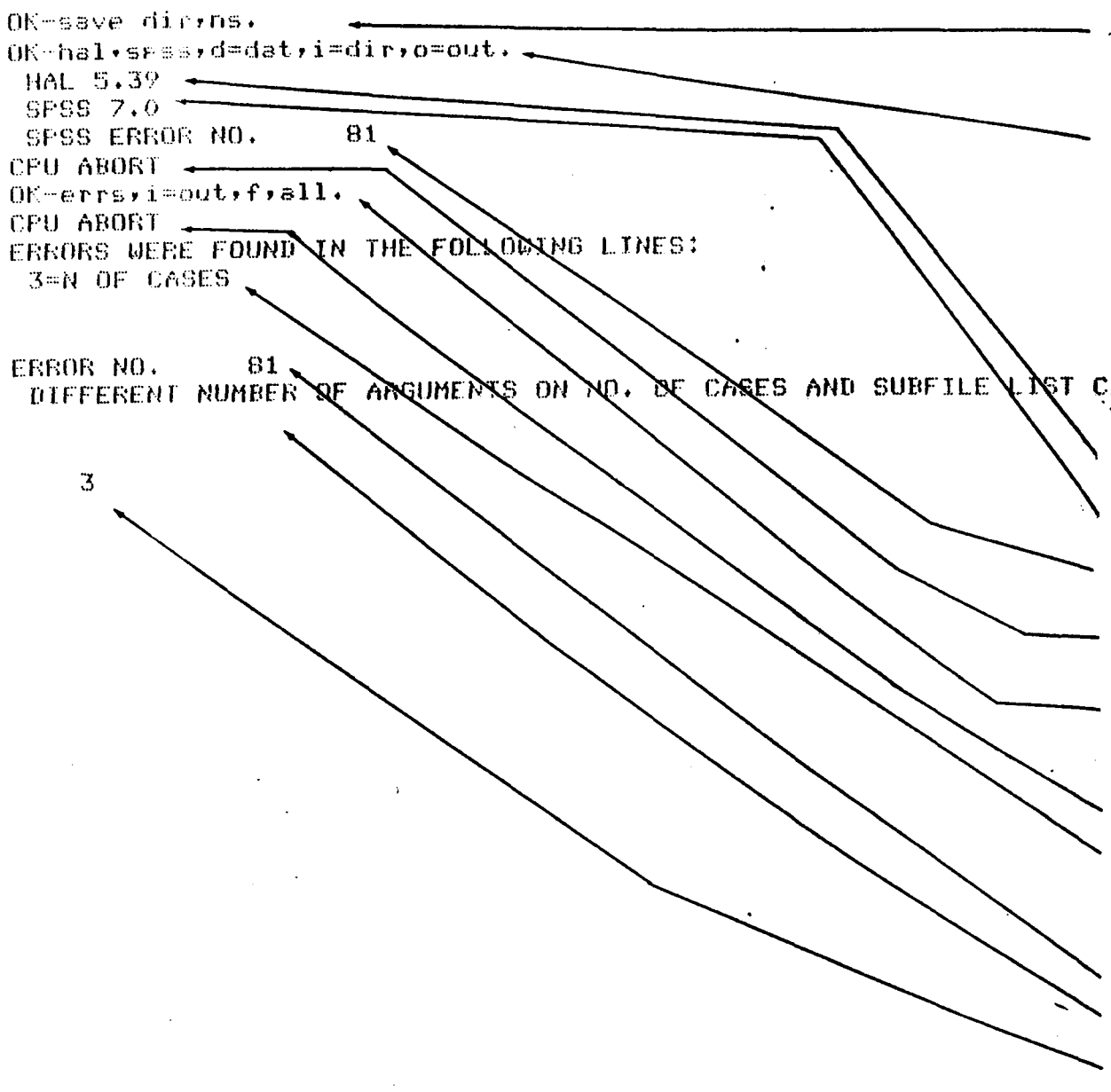
Request the BATCH editing system. System BATCH allows you to prepare a complete SPSS job for either batch or interactive processing. This example illustrates an interactive SPSS job. SPSS normally expects its program directives in 80 column format; this is the default for SYSTEM BATCH, so you don't need to specify length separately. The specification field on the SPSS program directive must begin in column 16 using the TAB directive (3.6.3). The TABCH directive (3.6.4) sets the tab character (in this case, semicolon). Note that this statement could be omitted because *semicolon is the default tab* character. You may choose any convenient character as a tab character (just be sure it does not normally appear in the text).

Request automatic line numbering beginning with line 100. Since no increment is specified, EDITOR will increment lines by 10 (3.7.1).

Your SPSS program directives. Note the semicolon (;) indicating a tab. This was the tab character defined above. Also, in line 130 notice how the backspace was used to correct the typing error in the word "values."

Typing an equal sign (=) followed by a carriage return indicates you have finished entering lines into EWFILE (3.7.1).

The text is being processed by EDITOR.



EWFILE is copied (using the SAVE directive) to a standard coded file with local file name, DIR, so that it can be processed by the system. NS indicates the file will have no sequence numbers (i.e. the EWFILE line numbers are not retained) (3.8.2).

Calls the SPSS program: the D parameter indicates the data file, DAT, which was attached in above. The input file, I, indicates the local file name of the file containing the SPSS program directives, DIR, which is your saved EWFILE. The output file, O, is assigned the local file name onto which the printed output from the job will be written (OUT). (*SPSS-6000 Supplement* 1.2)

The HUSTLER Auxiliary Library on which SPSS resides. In this case the current version is 5.39.

The current version of SPSS (in this case version 7.0).

An error has been detected during execution of the SPSS program.

Since an error was found, the job aborts (i.e. processing is discontinued.)

ERRS, an error scanning program is called. The input file for the ERRS program is your SPSS outfile OUT. The F parameter indicates search for full errors. (6.1.1)

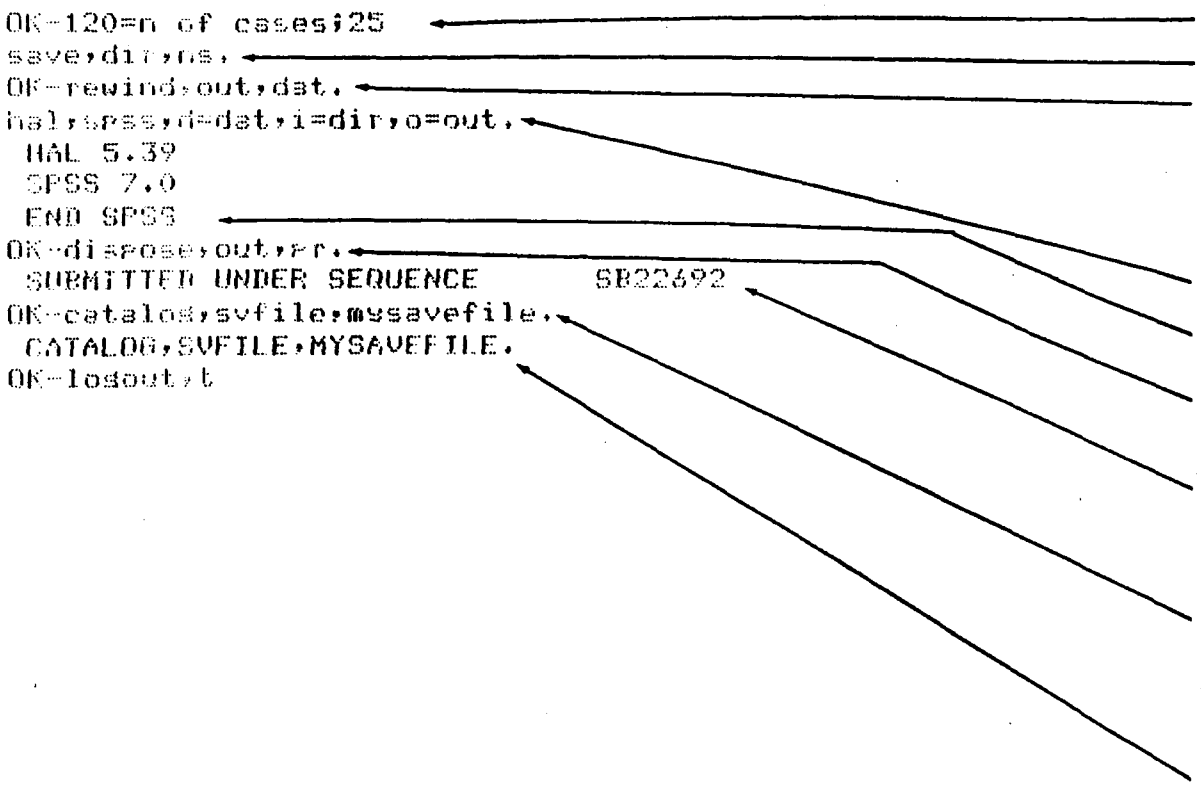
Messages from the ERRS program.

An error was found in the third line down of your directive file; the contents of that line are echoed. (The number is not the EWFILE line number. The error occurs in the third line, as counted by the SPSS program.)

The SPSS error number is 81.

Definition of error 81.

Line number of your file containing the error is repeated.



Correct the error by retyping the line.

Save the corrected EWFILE so that the new information can be given to SPSS.
Rewind the output and data files so that they are positioned at the beginning. If you use a file during an interactive session, be sure to rewind it before you use it again. Note that it was not necessary to rewind DIR, because the SAVE directive does that automatically. (Chapter 7, *SCOPE/HUSTLER Reference Manual*)

Call the SPSS program again.

Indicates that the SPSS program has finished execution.

Dispose (send) the listing of the job and the output to a line printer using the DISPOSE control statement (7.1).

Sequence number of the job (look for this number — last three digits — in the output bins on the 2nd floor). (Chapter 3 *SCOPE/HUSTLER Reference Manual*)

Catalog the SPSS SAVE FILE you created in line 12, as permanent file MYSAVEFILE. SPSS automatically writes its SAVE FILE to the local file, SVFILE. So, SVFILE, is the local file name used on the CATALOG statement. (Chapter 5, *SCOPE/HUSTLER Reference Manual*)

Echo of your CATALOG control statement which indicates that the temporary file has been cataloged as a permanent file.

Index

Comment Sheet

TITLE: Interactive System User's Guide

REVISION: K

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: _____

No postage stamp necessary if sent through campus mail. Fold on dotted lines and staple.

Staple

Staple

Fold

Fold

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

Fold

Fold