Price: $4.00

TYMSHARE TYMCOM-X

XEXEC

REFERENCE MANUAL

TYMSHARE, INC.
CUPERTINO, CALIFORNIA

TYMSHARE®

FEBRUARY 1978
VERSION 2

CONTENTS

Section 1
INTRODUCTION

XEXEC is the key to the TYMCOM–X system. XEXEC activates all the programs and languages on the system through user commands. In addition, XEXEC commands provide an effective and flexible file and file security system. XEXEC commands are simple and easy to use.

The XEXEC user has at his disposal a complete set of commands to handle all his activities on the TYMCOM–X, including

- Entering and leaving the Tymshare system
- Creating, modifying, deleting, and renaming files
- Setting file access controls
- Running programs
- Determining terminal connect time, machine number, etc.
- Initiating detached processing
- Executing commands indirectly

SYMBOL CONVENTIONS

The symbols used in this manual to indicate user-typed Carriage Returns and Alt Mode/Escapes are

Carriage Return: ↵

Alt Mode/Escape: ⊕

To indicate clearly what the computer prints and what the user types, all user-typed information is underlined. For example, in the lines

*WRITE↵
TO: TIMES↵

the computer prints an asterisk (*) and the user types WRITE followed by a Carriage Return. Then the computer prints TO: and the user responds by typing TIMES followed by a Carriage Return.

Lowercase letters in a command form represent the type of information to be entered. For example, the characters *file name* in the command form

–GO file name↵

indicate that the user types the name of a file at that point.

Braces in a command form indicate that the user must enter one of the items enclosed in the braces. The braces are not part of the command. For example, the command form

–{TYPE / PRINT / LIST} file list↵

indicates that the user precedes the file list with one of the words TYPE, PRINT, or LIST.

Brackets in a command form indicate an option; they are not part of the command. For example, the command

-MODIFY [file name]↵

indicates that the user may optionally specify a file name.

Control characters are denoted by a superscript *c*. For example, Z^c denotes Control Z. A control character typed by the user is shown in italics and is underlined. Control characters do not print on the terminal but are shown in examples for clarity. For example,

-DIRC*A*^{c}\C\ECTORY↵

indicates the following activity: the user types DIRC, then types a Control A to delete the letter C; the system prints \C\; then the user types the rest of the word and a Carriage Return.

The method of typing a control character depends upon the type of terminal being used. Consult the literature for your terminal or see your local Tymshare representative for information about typing control characters.

ABOUT THIS MANUAL

The rest of this section discusses the use of the Tymshare TYMCOM–X system. It explains how to make the connection to the Tymshare network via the user's terminal and how to log in, identify the terminal, and log out.

Section 2 provides a general description of XEXEC commands; it also describes the user's interaction with the system including how to enter information, the use of line-editing features, and the methods for declaring special terminal properties.

Section 3 explains the use of program and data files. The first part outlines the rules for naming files and notations that may be used when referring to files in commands. These notations allow the user to refer to many files with one expression, to exclude individual files when referring to a group of files, and to refer to files in another user's storage area. This section also contains a short description of Tymshare's EDITOR. The commands presented are adequate for the user to create and edit any file with ease. Finally, the section includes the commands for printing, duplicating, renaming, deleting, and comparing files.

Section 4 discusses the user file directory and file security controls. It explains the procedure for establishing a given level of security for individual files and for the entire file directory. This section also presents the commands used for obtaining information about the file directory. Finally, the switches that may be used to control the listing of directory information are presented.

Section 5 describes the EDITOR and conversational languages available on the TYMCOM–X and tells how to activate them. This section also discusses the use of applications programs available to XEXEC users.

Section 6 presents the commands and the various options for running user-written programs from XEXEC. Most of the options direct the compilation and execution of the user's program; for example, the user may specify libraries to be loaded with the program or may request a cross-reference listing. This section also discusses the extended command forms which are useful when working with many programs at one time, as well as the debugging aids available for user-written programs. Finally, Section 6 describes how to execute programs independently of the terminal.

Section 7 details the commands provided for listing information about the system and the user's

job. These commands display such information as the amount of unused core available, the current time, and the current job number. This section also presents the commands that change device assignments and the commands that transfer control from one operating mode to another.

Section 8 describes two methods of indirectly executing commands. Command string files allow the user to execute certain commands for a series of files; the PERFORM command allows the user to execute a sequence of commands stored on a file.

Appendix A contains an alphabetized list of all XEXEC command forms with a short description of each command.

Appendix B contains a table describing the terminal filler classes.

ENTERING AND LEAVING THE SYSTEM

Entering the Tymshare system requires three steps: calling the Tymshare network (TYMNET), identifying the type of terminal, and identifying the user. The name of this process is *logging in.*

Calling the Tymshare Network

The specific procedure for contacting the Tymshare network depends upon the terminal configuration. Most terminals communicate through an acoustic coupler or a data phone. This subsection describes the procedure for using these two devices.

Acoustic Coupler

1. Plug both cords from the terminal into the acoustic coupler, and plug the coupler into a standard three-prong wall outlet.
2. Put the terminal in the on-line or compute mode.
3. Using a regular telephone, dial the local TYMNET access number.
4. When a high-pitch tone sounds, place the telephone handset into the coupler with the telephone cord on the side indicated.
5. Push the ORIGINATE button on the acoustic coupler.

Data Phone

1. Put the terminal in the on-line or compute mode.
2. Push the TALK button.
3. Dial the local TYMNET access number.
4. When a high-pitch tone sounds, push the DATA button and replace the handset.

Identifying the Terminal

As soon as the user makes the connection to the Tymshare network, the system activates the terminal and sends a message. It transmits the message at 10 characters per second, which prints

legibly only on 10-character-per-second terminals. On other speed terminals, a sequence of usually unintelligible characters prints, and then the terminal pauses.

The user must first type the identification character for his terminal. This character tells the system which code and transmission speed to use in communicating with the terminal.

The table below lists the terminal identification characters. If the user has a question about which one applies to his particular terminal, he should contact his Tymshare representative.

CPS (In/Out)	Identification Character	Tymshare Model Number	Other Terminals	Comments (Unless noted, ASCII and no parity assumed)
10	D	200, 201	Teletype Model 33	
15	B		Teletype Model 37 (without parity)	
15	Carriage Return		IBM 2741, Datel/UCC, Itel/Dura, Novar, Trendata, AJ 841	Correspondence or EBCD code, depending on the telephone number called
15	J		Teletype Model 37 (with even parity)	ASCII (even parity)
15/30	F		Syner-Data Beta	Terminal must be equipped with dual-speed input/output
30	A	400	CRT Terminals	No Carriage Return or Line Feed delay
30	C	300, 301, 310, 311	Univac, Syner-Data Beta and Totalterm, Gulton	
30	G	321	Memorex 1240 and 1280, TermiNet 300	Delay after Line Feed
30	E	100, 110, 211, 212, 213, 221, 1030	Execuport, Texas Instruments Models 725 and 733, NCR 260 Series	Short Carriage Return delay

After the system receives the identification character, it prints the TYMNET node and port number followed by the request

PLEASE LOG IN:

Logging In

The log-in procedure requires typing a user name and a password which are both registered by Tymshare. A program called the Network Supervisor checks both the user name and the password before admitting the user to the computer system. In addition to the user name and password, the user may also specify a project code and a job number during the log in. The project code is useful when assigning costs since the billing information sent to each customer includes the project code. The job number is specified if the user wants to resume a job which had been previously detached.[1]

1 - Detached processing is described on page 67.

After the system prints

PLEASE LOG IN:

the user enters his name, password, and an optional project code. The user may also enter an optional job number at this point, to attach the terminal to a specified job.[1] The most efficient procedure is to type all entries on the same line, separating them with semicolons (;). In the example below, the user types his user name, MOORE, a password, and the project code, K-333, separating the entries by semicolons. A Carriage Return completes the log-in procedure. The system then prints the time and date, returns the carriage, and prints a prompt (–) at the beginning of the line.[2] For example,

```
-1002-21--

PLEASE LOG IN:  MOORE;;K-333↵

TYMSHARE  1116  29-APR-1974

-
```

The dash indicates that the user is in XEXEC and that the system is ready to accept a XEXEC command.

An alternative method of logging in is to type a Carriage Return after each entry and wait for the system to request the next entry. For example,

```
-1002-21--

PLEASE LOG IN:  ANDREW↵          The user enters his user name.

PASSWORD:  ↵                     The password is not printed.
PROJ CODE:  ↵                    The user does not choose to enter a project
                                 code, so he types a Carriage Return only.
TYMSHARE  1120  29-APR-1974

-
```

If the user makes an error while logging in, the system replies with

ERROR, TYPE

followed by another request for the information. In the following example, the user types an unacceptable user name, corrects his error, and continues to log in.

1 - Detached processing is described on page 67.
2 - If the user is validated on a TYMCOM–X system other than XEXEC, the prompt for that system is printed rather than a dash (–).

```
-1002-21--

PLEASE LOG IN:  MORE↵

ERROR, TYPE USER NAME:  MOORE↵

PASSWORD:  ↵
PROJ CODE:  K-333↵

TYMSHARE  1210  30-APR-1974

-
```

The user can correct an error made while typing his project code by typing a Control A to delete the previous character. Each additional Control A deletes one additional character. For example, the system accepts the following project code as THERMAL

PROJ CODE: THAR*A*^{C}\R*A*^{C}A\ERMAL↵

Note that Control A prints a back slash (\) followed by each character as it is deleted. When the user types a character other than Control A, the new character is preceded by a second back slash, thereby enclosing all deleted characters within a pair of back slashes.

NOTE: Control A cannot be used to correct a user name or password.

The error messages are the same regardless of which log-in procedure the user follows.

Associated with every user name is a specific computer system referred to as the user's home system. If a user is validated on more than one computer system, the log-in procedures described previously connect him with his home system. To log into a system other than his home system, the user types a colon (:) and the system number after his user name. For example, user HANSEN is validated on system 2 and system 34; system 2 is his home system. To log into system 2, he types his user name and password as shown below:

PLEASE LOG IN: HANSEN;;↵

To log into system 34, he types his user name and password as shown below:

PLEASE LOG IN: HANSEN:34;;↵

The Network Supervisor allows two minutes to log in. Tymshare sets this time limit to prevent an unauthorized user from accessing the system. If the user does not complete the log in within the time limit, the Network Supervisor prints a disconnect message and terminates the connection.

If the user has difficulty logging into the system or receives no response, he should call the local Tymshare office. Local TYMNET telephone numbers are usually posted on the terminal.

Logging Out

To leave the system, the user types LOGOUT, or simply LOG, followed by a Carriage Return. The BYE and KJOB commands also perform this function. In response, the system prints the Tymshare Resource Units (TRUs) and terminal connect time used.

```
0.89 TRU                          TRUs consumed.
TERMINAL TIME: 0:00:14            Connect time.

PLEASE LOG IN:
```

The EXIT command may also be used to leave the system. It is the same as LOGOUT except that the system does not print the number of TRUs consumed or the terminal connect time.

After leaving the system, the user disconnects the terminal, logs into another system, or lets another user log in. The next user should not retype the terminal identification character when logging in on the same network connection.

If the terminal remains connected to the Network Supervisor and a user does not log in within two minutes, the Network Supervisor prints a disconnect message and terminates the telephone connection.

Section 2
USING XEXEC

This section provides an overview of XEXEC describing general information about XEXEC commands. The user is introduced to XEXEC commands and shown how to enter information at the terminal, correct errors made while entering information, and declare special terminal properties.

THE FORM OF XEXEC COMMANDS

The different elements of a XEXEC command are separated from each other by spaces. For example,

-RENAME FILE1 AS FILE2↵

The elements of a command need not be separated by spaces if no confusion is caused by omitting the spaces. For example, the space between the file name NEWPR and the switch name /SMALL can be omitted in the command

-TYPE NEWPR/SMALL↵

but not in the command

-TYPE /SMALL NEWPR↵

Many XEXEC commands have the form

-command file list↵

where file list is a list of file designations and possibly switches which modify the command.[1] The format of the file list is governed by the following rules:

- Multiple file names are separated by commas.
- Multiple switches are optionally separated by spaces.
- A switch that immediately follows a file name need not be separated from the file name by a space.
- A switch that precedes a file name must be separated from the file name by one or more spaces.
- A switch that precedes a list of file names applies to all the file names that follow the switch.
- A switch that immediately follows a file name applies to that file name only.
- A switch that immediately follows a file name takes precedence over a switch that precedes a list of file names.

REQUESTING INFORMATION ABOUT XEXEC COMMANDS

To obtain a list of XEXEC commands, the user simply types

-HELP↵

1 - Command switches are described on page 55.

To obtain a list of options or switches available for a particular command, the user types the command name followed by /HELP or by /?. For example,

```
-FILES /?↵
OPTIONS ARE:LPT,EXTENS,REVERS,ALPHAB,SIZE,CREATI,TIME,SECOND,TOT
,PROTEC,ACCESS,MODE,TEMPS,K,FAST,EVERYT,UNSORT,STORAG,UFDBIT,NOP
,STATUS,BEFORE,AFTER,TODAY,WAIT,RUN.

-
```

USER INTERACTION

The procedures for communicating with the system include entering information at the terminal, editing information as it is entered, and declaring special terminal properties. These procedures are discussed below.

Entering Data at the Terminal

Throughout a session at the terminal, the system prompts the user whenever it is ready to accept a command, a response, or any user input. XEXEC prompts the user with a dash (–) to indicate that it can accept a XEXEC command. The user may abbreviate any XEXEC command to the fewest leftmost characters required to uniquely identify the command.

NOTE: As new commands are added to XEXEC, the number of characters required to uniquely identify a command may change.

If XEXEC does not recognize a command, it responds by printing what the user typed surrounded by question marks. For example,

```
-NOCOMMAND↵
?NOCOMMAND?
-
```

Subsystems of XEXEC prompt the user with a character other than a dash.[1] The example below illustrates the user's interaction with the system as he calls EDITOR and enters an EDITOR command.

```
-EDITOR↵          XEXEC prints a prompt and the user enters a XEXEC command.
*READ INVEN↵      The subsystem prints a prompt and the user enters an EDITOR command.
1980 CHRS
*                 The subsystem prompts the user for another EDITOR command.
```

Line-Editing Features

The user can edit information being entered by deleting a character or by deleting an entire line. Control A deletes the preceding character; it may be used repeatedly to delete several preceding characters. The system surrounds the character(s) deleted with back slashes (\). For example, the line

1 - XEXEC subsystems are discussed on page 45.

-DELTEA^C\E$A^C$T\ETE PROG1↵

is accepted as

-DELETE PROG1↵

Control Q deletes an entire line of user input. After the user types a Control Q, the system prints an up arrow (↑) followed by a Carriage Return and waits for the user to reenter the line. For example,

-TYPE PG1, PG2Q^C↑
TYPE PG1, PG2↵

If the user types an Alt Mode/Escape at any time before the terminating Carriage Return, the command is aborted.

NOTE: When a program is waiting for input from the terminal, one Alt Mode/Escape returns control to XEXEC. Unless a program is designed so that only a direct command terminates execution, two consecutive Alt Mode/Escapes terminate current operations, abort any commands that have not been executed, and return control to XEXEC.

Declaring Special Terminal Properties

The user may declare the special properties of his terminal with the TTY command. The TTY command can be used to simulate features such as tabs and form feeds on terminals not equipped with these functions.

The table below explains the various commands for special terminals, their meanings to the system, the standard state of TYMCOM–X terminals, and the procedures which normally take place.

Special Terminal Command	Meaning	Standard State	Meaning
TTY TAB	This terminal is equipped with TAB stops; when a tab appears in text, the system sends Control I	TTY NO TAB	The system simulates the TAB output by sending an appropriate number of spaces instead of Control I
TTY FORM	This terminal is equipped with FORM (page) and VT (vertical tab) characters. The system sends these characters without simulation	TTY NO FORM	The system simulates FORM with eight Line Feeds, and VT with four Line Feeds
TTY FILL *n*	The terminal is assigned filler class *n*.[1] The filler character is DEL or RUBOUT	TTY NO FILL	The filler class is 0. All fillers are equal to 0

(Table continues)

1 - The filler classes are described in Appendix B on page 91.

Special Terminal Command	Meaning	Standard State	Meaning
TTY LC	The system accepts lowercase characters and does not convert them to upper case	TTY NO LC	The system translates lowercase characters entered by the user to uppercase characters
TTY WIDTH *n*	The width of the carriage is set to *n*, which determines the number of characters per line preceding the automatic Carriage Return at the end of the line. The value of *n* may range from 17 to 200	TTY WIDTH 72	The carriage is 72 characters wide with an automatic Carriage Return inserted after column 72
TTY NO CRLF	The system suppresses the automatic insertion of a Carriage Return when the carriage exceeds its width	TTY CRLF	Restores the normal insertion of Carriage Returns
TTY NO ECHO	This suppresses the normal echo sent from the system after receiving input from the user	TTY ECHO	Restores the normal procedure of echoing each character after it is received

Section 3
USING FILES IN XEXEC

Files are an important part of the Tymshare system. A file contains information such as a program or data and is stored on a storage device. Unless the user declares files as temporary, files are permanent storage; that is, they stay on the system until the user explicitly removes them, even though he logs off after each session at the terminal. XEXEC has commands to create and modify files, list descriptive information about the files, change the standard protection controls for files, copy files, print files, rename files, and delete files from the storage area.

RULES FOR NAMING FILES

The user identifies a file with a file name and an optional extension. A file name may contain as many as six characters, with any combination of letters and digits. The following are acceptable file names:

PGM3 LDATA 3TEST SURVEY FPROG

The user may find it convenient to include an extension in the file identifier to indicate the type of information stored on the file. Usually the extension is a standard one which XEXEC recognizes. A period (.) separates the file name from its extension. The extension may contain no more than three alphanumeric characters. Some standard extensions and their meanings are listed on the following page. The following list of file names with standard extensions demonstrates the contents of a typical user's storage area:

STAT.F4	**STAT.SAV**	**ACCTG.SAV**
ROUT3.CRF	**POLL.DAT**	**ROUT3.REL**
PGM1.REL	**ROUT3.MAC**	**PGM1.F4**

A list of standard XEXEC extensions and their meanings is presented below.

Extension	Meaning to the System
F4	FORTRAN IV source program
FTF	FORTRAN source program
MAC	MACRO source program
BAS	BASIC source program
CBL	COBOL source program
BLI	BLISS source program
FAS	FASBOL source program
SIM	SIMPLE source program
FAL[1]	FAIL source program
SAI[1]	SAIL source program
ALG[1]	ALGOL source program
LSP[1]	LISP source program
SNO[1]	SNOBOL source program
VAS	Varian assembler source program
REL	Relocatable binary file that the compiler/assembler produces
SAV	Executable file that the SAVE command produces and the RUN command uses
HGH[2]	Executable file containing the high segment of a two-segment program produced by the SAVE command
LOW[2]	Executable file containing the low segment of a two-segment program produced by the SAVE or SSAVE command
SHR[2]	Executable file containing the sharable high segment of a two-segment program produced by the SSAVE command
LST	Ordinary listing
CRF[3]	Cross-reference data which the CROSS command processes
CMD[4]	File containing text to be used as part of a command
DAT	Data file
TMP	Temporary file which the system automatically deletes when the user logs out

1 - These extensions refer to language compilers that are available on the system although Tymshare does not currently support these languages.

2 - See the *DECsystem10 Assembly Language Handbook* for further explanation.

3 - The user creates the CRF file using the /CREF switch, discussed on page 56; the CROSS command, discussed on page 61, processes the CRF files.

4 - See the @ construction on page 81.

NOTE: The source program or data file name may have the null or blank extension. For example, the following names have the null extension:

FPROG	*PGM2*	*CDATA*	*TEXT2*

The system reserves certain file names which the user should not attempt to use to identify his files. The following are reserved file names: ALL, HELP, NOT, SAME, and any left subset of the words TERMINAL and TELETYPE.

REFERRING TO FILES IN COMMANDS

All XEXEC commands recognize a reference to an existing file if both the file name and extension are specified. Certain XEXEC commands allow the user to refer to a file without specifying an extension provided that the file identifier has a standard extension or no extension. The compile-type commands, described on page 51, search for a standard extension or append a standard extension to a file identifier that has no extension.[1] The GET, RUN, and GO commands, described on page 66, search for the HGH extension first, then the SHR extension, then the SAV extension. If the file identifier has no extension, SAV is assumed. If a file identifier has a non-standard extension, the extension must be specified when referring to the file.

Asterisk (*) Notation

Asterisk notation allows the user to specify several files with a single identifier. An asterisk (*) can be used as either part of a file identifier to indicate all file names or all extensions. For example, TEST1.* refers to all files with the name TEST1, regardless of the extension. Similarly, *.F4 refers to all files with the F4 extension. The user may refer to all files in his directory with *.* or the word ALL.

Asterisk notation can be used with the following commands: COPY, DECLARE, DELETE, DIRECTORY, FILES, LIST, PRINT, RENAME, and TYPE.

Crosshatch (#) or Question Mark (?) Notation

Crosshatch or question mark notation also allows the user to specify several files with a single identifier. A crosshatch (#) or a question mark (?) can be used instead of a character in a file identifier to mean any character. The crosshatch or question mark can be used repeatedly and can be interspersed throughout the file identifier. For example, #P##.F4 could refer to CP12.F4, APPS.F4, etc. If the crosshatch or question mark is at the end of a file name or extension, it represents no character as well as any character. TEST#.REL, for example, refers to all file identifiers with the REL extension and the file name TEST followed by one character or by no characters as in the case of TEST.REL. Consider the file identifiers listed below.

TEST	**TEST1.SAV**	**TESTA.NEW**
TEST1.F4	**TEST2.F4**	**TESTA.REL**
TEST1.REL	**TEST3.SAV**	**TESTS**

1 - The user should specify the extension if he has more than one file with the same name but with extensions that imply different compilers, for example, TEST.F4 and TEST.CBL.

The user may refer to all the files previously listed by typing

TEST#.*

To refer to all TEST1 files, the user types

TEST1.*

To refer once to all the above files that have the NEW or REL extension, the user types

TEST#.#E#

The crosshatch or question mark notation can be used with the following commands: COPY, DECLARE, DELETE, DIRECTORY, FILES, and RENAME.

NOT or Minus Sign (-) Notation

To exclude a particular file or group of files from a command, the user may type NOT or a minus sign (-) followed by the file name or group of file names to be excluded. For example, to refer to all files except those with the .F4 extension, the user types

ALL NOT *.F4

To refer to all files that begin with TEST, except TEST4 and TEST5, the user types

TEST#.*-TEST4.*-TEST5.*

The NOT or minus sign notation can be used with the following commands: DECLARE, DELETE, DIRECTORY, and FILES.

Referring to Files in Another User's Storage Area

A user can refer to a file in another user's storage area by preceding the file name with the owner's user name, enclosed in parentheses. For example, (OLSON)INVEN refers to the file INVEN located in OLSON's storage area. One user may gain access to a file in another user's storage area if the owner declares that file to be sharable.[1] The owner can allow another user to run programs from his storage area, read files from his storage area, or change files in his storage area.

For example, if user WONG declares his file COMPUTE to be sharable, user ALVIN can copy COMPUTE to a file named MYCOMP which is in ALVIN's storage area by typing[2]

-COPY (WONG)COMPUTE TO MYCOMP↵

In the FILES, DIRECTORY, DELETE, and DECLARE commands, only the first file name in a list of file names can have a user name associated with it. This user name applies to all the files in the list. For example,

-DELETE (MARGE)FIL1,FIL2,FIL3↵

deletes FIL1, FIL2, and FIL3 in MARGE's storage area if the protection of each file has been declared ALL.

1 - The DECLARE command is described on page 37.
2 - The COPY command is described on page 22.

INTRODUCTION TO EDITOR

EDITOR allows the Tymshare user to create or edit any file easily and quickly. The file may contain text, a computer program in any language, or symbolic data which the user wants to enter into the system at a later time. The user may enter information into EDITOR from a file or directly from the terminal.

The user calls EDITOR from XEXEC by typing

-EDITOR↵

Then EDITOR prompts with an asterisk (*) and waits for the user to enter a command.

The *Tymshare EDITOR Reference Manual* and the *Tymshare Addendum to EDITOR* thoroughly describe EDITOR's convenient commands and elaborate capabilities. This discussion is simply an introduction to the most popular features for creating and editing files.

Creating a File

Most files are created in EDITOR. To create a file, the user must first enter the contents of the file as lines of text using the APPEND command. He first types

*APPEND↵

Next, the user types as many lines of text as desired. Each line is terminated by a Carriage Return. To quit entering text, the user types a Control D as the first character of a line.[1] EDITOR responds by printing an asterisk prompt character. The user may then type the WRITE command to save on a file the text just entered. The command form is

*WRITE file identifier↵

or simply WRITE followed by a Carriage Return. When the user types WRITE only, EDITOR prompts for a file identifier on which to store the data. For example,

```
-EDITOR↵
*APPEND↵
98.6 99.0 98.7 100.0↵
99.0 98.6 98.5 99.0↵
99.6 98.6 101.0 105.0↵
Dc                          The user types a Control D to terminate the APPEND
*WRITE↵                     command and then enters the WRITE command.
TO: TDATA↵
 NEW FILE↵
70 CHRS
*QUIT↵                      The QUIT command returns control to XEXEC.

-
```

1 - If a Control D is typed in midline, it adds the remaining characters of the previous line onto the current line; it does not terminate the APPEND command.

Whenever the user types the WRITE command, EDITOR checks to see whether a file with the identifier specified already exists in the user's storage area. The system prints NEW FILE if the user types a new file name, or OLD FILE if a file name already exists. Then, EDITOR waits for the user to confirm or abort the WRITE command. A Carriage Return confirms the WRITE command. If the user types a Carriage Return after the OLD FILE message, his current text replaces the contents of the old file. The user may abort the WRITE command and save the contents of the old file by typing any character other than a Carriage Return; EDITOR automatically returns the carriage and allows the user to enter another command. The example shown below illustrates this feature.

```
-EDITOR↵
*APPEND↵
THIS IS RANDOM TEXT↵
Dc
*WRITE TDATA↵
 OLD FILE N↵                    The user types an N to abort the command. He may enter
*                               the WRITE command again with a different file name.
```

The example below demonstrates the use of a few control characters to perform editing functions while creating a file. Control A deletes the preceding character and prints a back slash (\) and the deleted character. The user may type it repeatedly to delete several characters; each repeated use prints the deleted character. Control Q deletes the current line and prints an up arrow (↑). Control W deletes the preceding word, defined as the immediately preceding blanks, if any, plus the immediately preceding nonblank characters, up to but not including the first blank preceding them; the deleted characters are printed surrounded by back slashes.

```
-EDITOR↵
*APPEND↵
ABCDFAc\F\EFGH↵                 Control A deletes F.
IKJLMNOPGQc↑                    Control Q deletes the line and returns the carriage.
IJKLMNOPQ↵
RSTUVWXYZ↵
THE ALPHBETWc\TEBHPLA\ALPHABET↵ Control W deletes ALPHBET.
Dc                              Control D terminates the APPEND mode.
*/                              The system prints the prompt character, and the user
ABCDEFGH                        requests a listing of the current text by typing a slash.
IJKLMNOPQ
RSTUVWXYZ
THE ALPHABET
*WRITE ALPHA↵                   The user enters the command to write the text on a
 NEW FILE↵                      file named ALPHA.
50 CHRS
*QUIT↵                          The QUIT command returns control to XEXEC.

-
```

The slash (/) can be used to print all or part of the current text in EDITOR. To print all the text, the user simply types a slash. To print a line or a range of lines, the user types

*r/

where r represents a line, or a range of lines. The range is indicated by the numbers of the first and last lines; the numbers are separated by a comma.

For example,

```
*3,4/
```

prints lines 3 and 4.

Editing a File

The user may change the contents of an existing file using EDITOR. First, he must instruct EDITOR to read the file into the text area. He accomplishes this with the READ command. For example,

```
-EDITOR↵
*READ EXAMPL↵
65 CHRS
*/                    The user types a slash to print the contents of the file.
124456789
400 200 300 400 500
THERE IS A MUSTAKE IN THIS LINE
*
```

The user may make changes within a line of text with the EDIT command. The command form is

```
*r EDIT↵
```

where *r* represents a line or a range of lines which the user wants to edit; a range of lines is indicated by typing the first and last line of the range, separated by commas. For example, to edit lines 1 through 5, the user types

```
*1,5 EDIT↵
```

The user edits the first line of the example printed above:

```
*1 EDIT↵              EDITOR prints the line to which the EDIT command refers and returns the carriage.
124456789
123456789↵            The user retypes the line, making the changes desired.
*
```

The *Tymshare EDITOR Reference Manual* describes many additional editing capabilities and conveniences. This subsection presents an explanation of two control characters that may be used in editing a line.

Control D copies the remainder of the old line after the user makes changes to the beginning of the line. In this example, the user wishes to edit line 2, changing the first 4 to a 1. First, he specifies the line number and types the EDIT command. EDITOR prints line 2.

```
*2 EDIT↵
400 200 300 400 500
1Dc00 200 300 400 500   The user types a 1 and then Control D, which
*                       copies the rest of the old line to the new line.
```

Control Z copies characters from the old line up to and including the character typed after it. For example, the user wants to edit line 3.

```
*3 EDIT↵
THERE IS A MUSTAKE IN THIS LINE
ZcMTHERE IS A MIDcSTAKE IN THIS LINE
*/
123456789
100 200 300 400 500
THERE IS A MISTAKE IN THIS LINE
*
```

Control Z followed by an M copies the old line through the first occurrence of M. Next, the user types the correction followed by a Control D, which copies the rest of the line and returns the carriage.

The DELETE command removes one or more lines from a file. The command form is

*r DELETE↵

where *r* represents a line or range of lines. In the example below, the user deletes line 3. Next, he adds two lines to the end of the text. He accomplishes this with the APPEND command, discussed on page 17.

```
*3 DELETE↵
*APPEND↵
13579↵
115533↵
Dc
*/
123456789
100 200 300 400 500
13579
115533
*
```

A Control D typed immediately after a Carriage Return terminates the APPEND mode. The user types a slash to examine the current text.

Whereas the APPEND command adds additional lines at the end of the text, the INSERT command adds additional lines anywhere in the text. In the following example, the user inserts two lines before line 2.

```
*2 INSERT↵
ABCDEF↵
GHIJKL↵
Dc
*/
123456789
ABCDEF
GHIJKL
100 200 300 400 500
13579
115533
*WRITE↵
TO: EXAMPL↵
 OLD FILE↵
65 CHRS
*QUIT↵
```

A Control D typed immediately after a Carriage Return terminates the INSERT mode. The user types a slash to examine the current text.

The user writes the edited text on the same file.

He confirms the WRITE command, which deletes the previous contents of EXAMPL.

The QUIT command returns control to XEXEC.

The QUIT command returns control to XEXEC. The user can also return control to XEXEC by typing one or more Alt Mode/Escapes. If the user inadvertently returns control to XEXEC by typing Alt Mode/Escapes before he writes his file, he should immediately type the CONTINUE or REENTER command to return control to EDITOR.[1] The user will find his text intact, with the possible exception of the last line typed. If the user attempts to leave EDITOR by typing QUIT before he writes his file, EDITOR warns him with the message

FILE NOT WRITTEN – OK?

An N or NO answer leaves control in EDITOR. A Carriage Return or Y returns control to XEXEC and the edited text is lost.

Summary

The tables which follow summarize the information presented in this section.

EDITOR Command	Action
READ file name	Reads the specified file into the text area. The command is terminated with a Carriage Return
APPEND	Adds the following lines to the end of the current text. The entry is terminated with a Control D
WRITE file name	Stores the current text on the specified file. The command is terminated with a Carriage Return
r EDIT	Prints the specified range of lines one line at a time and allows the user to change them. The user edits each line and the edited lines replace the old lines. The command is terminated with a Carriage Return
r DELETE	Deletes the specified range of lines. The command is terminated with a Carriage Return
n INSERT	Inserts the following text before line *n*. The entry is terminated with a Control D
[*n*] /	Prints line *n* or all the current text if *n* is not specified. No Carriage Return is required after a slash.
QUIT	Returns control to XEXEC. The command is terminated with a Carriage Return

1 - See page 46 for an explanation of the CONTINUE and REENTER commands.

Control Character	Function
A^C	Deletes preceding character; it may be used repeatedly to delete several characters
D^C (after a Carriage Return)	Terminates the APPEND or INSERT command
D^C (within a line)	Copies the rest of the old line (in EDIT), or copies the rest of the preceding line (in APPEND or INSERT)
Q^C	Deletes the entire line
W^C	Deletes the preceding word, defined as the immediately preceding blanks, if any, plus the immediately preceding nonblank characters, up to but not including the first blank preceding them
Z^C (followed by a character)	Copies the previous contents of the line up to and including the character typed after Control Z (in EDIT), or copies the preceding line up to and including the character typed after Control Z (in APPEND or INSERT)

THE COPY COMMAND

The COPY command enables the user to create a new file, copy an existing file in his own or another user's storage area, or concatenate two or more files. The form of the COPY command is

-COPY source $\left\{\begin{matrix}\text{TO} \\ , \end{matrix}\right\}$ destination↵

where source and destination are file identifiers. The contents of the source file are copied and saved on the destination file. If the user simply types COPY followed by a Carriage Return, the system prompts for the file identifiers. For example,

```
-COPY↵
FROM FILE: OLDF↵
TO FILE: NEWF↵

-
```

Creating a File

The user may create a file directly in XEXEC with the COPY command by specifying the terminal as the source. For example,

-COPY TERMINAL TO NEWFIL↵

creates a file named NEWFIL, which contains the text entered after the COPY command. If NEWFIL already exists, the COPY command deletes the old contents of the file and replaces them with the new text.

NOTE: The word TERMINAL may be abbreviated to any left subset of the word.

After typing the COPY command, the user enters his text. Only two editing characters are available with the COPY command: Control A and Control Q, which are described on pages 10 and 11. The user types a Control D or a Control Z to terminate entering text and to create the file with the designated file identifier. For example,

```
-COPY T TO SDATA↵
25,11,21,46↵
21,5,25,50↵
57,5,6,14↵
56,4,1,15↵
.
.
.
Dc
-
```

Duplicating a File

The user may duplicate a file using the COPY command by specifying an existing file as the source. For example, to duplicate the contents of file ADDRS1 on another file named ADDRS2, the user types

```
-COPY ADDRS1 TO ADDRS2↵
```

The same text exists on two different files, ADDRS1 and ADDRS2.

The user may also copy the contents of a file to or from another user's storage area. This is possible only if protection of the file to be copied allows other users to read it. See page 37 for a discussion of file protection.

The form of the COPY command to copy a file to the current user's storage area is

```
-COPY (user name)file name TO file name↵
```

To copy a file to another user's storage area, the form is

```
-COPY (user name)file name TO (user name)file name↵
```

A destination file identifier of SAME assigns the new file the same name it had in the other user's storage area. For example,

```
-COPY (MOORE)LANI TO SAME↵
```

copies the contents of the file named LANI in user MOORE's storage area to a file named LANI in the storage area of the current user.

If the user specifies the terminal as the destination, the contents of a file are printed at the terminal. For example,

```
-COPY AMT TO TERMINAL↵
   457.39
   500.00
  1434.90
   892.47
  1553.79
   231.38

-
```

Concatenating Files

The user may concatenate the contents of two or more files and store the result on another file by using a plus sign (+) between the files to be concatenated. The appropriate form of the COPY command is

–COPY source$_1$+source$_2$+ · · · +source$_n$ {TO / ,} destination↵

For example, assume that file A contains the text

LINE 1 FROM FILE A

and file B contains the text

LINES 2 AND
3 FROM FILE B

The user types the command

–COPY A+B,C↵

File C now contains the text shown below.

```
-TYPE C↵
LINE 1 FROM FILE A
LINES 2 AND
3 FROM FILE B

-
```

This form of the COPY command can also be used to concatenate and copy files from another user's storage area. When a user name precedes a file in the list, it applies to all subsequent files joined by plus signs until a new user name appears in the list. For example,

–COPY (SFS)TEST+TEST1+(VIOLET)FILEA+FILEB,A↵

concatenates files TEST and TEST1 in user SFS's storage area and files FILEA and FILEB in user VIOLET's storage area, and writes the result on file A in the storage area of the current user.

NOTE: If the user specifies the word SAME as the destination, the files are not concatenated but are copied individually on files with the same name in the user's storage area. For example, the command

–COPY (SFS)TEST+TEST1 TO SAME↵

duplicates the contents of (SFS)TEST on a file named TEST in the user's storage area and duplicates the contents of (SFS)TEST1 on a file named TEST1 in the user's storage area.

DELETING FILES

The DELETE command removes a file or group of files from the user's directory. The command form is

–DELETE file list↵

where file list is a list of file names separated by commas and optional switches, including the /WAIT switch, described below, or any of the item, sort, or select switches listed on page 42; these switches provide additional information about the files, delete the files in a certain order, or select certain files to delete.

The system deletes the file only if its protection allows deletion,[1] that is, if its protection is ALL. If the user attempts to delete a file that has protection other than ALL, the system prints

file name **PROTECTED**

and does not delete the file. For example,

```
-DELETE TDATA↵

TDATA          **PROTECTED**
-
```

The /WAIT switch instructs the system to ask the user to verify the DELETE command for each file specified in the command. The system prints the heading

FILES DELETED:

followed by a file identifier and a question mark (?) for each file specified in the command. The user must type Y to delete the file and N to prevent the deletion of the file. A Carriage Return need not be typed after the response. For example, the user wants to delete the files INV.SEP, PGM1.REL, and PGM1.F4:

```
-DELETE /WAIT INV.SEP,PGM1.*↵

FILES DELETED:
PGM1     F4    ?Y
PGM1     REL   ?Y
INV      SEP   ?Y
PGM1     CRF   ?N **NOT DELETED**

-
```

In the example shown below, the user wants to delete specific files created after April 24. He uses the /WAIT switch and also requests that the system print the size with the file identifier.[2]

```
-DELETE ALL /WAIT /SIZE /AFTER 24-APR↵

FILES DELETED:
OLDF                5  ?Y
TDATA               5  ?Y **PROTECTED**
TEXT     LST       10  ?N **NOT DELETED**
TEXT                5  ?N **NOT DELETED**
A                   5  ?Y

-
```

Note that, even though the user types Y, the system does not delete the file because the file's protection prevents deletion.

1 - Refer to page 37 for a discussion of file protection.
2 - The /SIZE and /AFTER switches are discussed on pages 42 and 43.

RENAMING FILES

The RENAME command changes the identifiers of existing files. The command form is

-RENAME old file identifier $\left\{\begin{matrix}\text{AS} \\ , \end{matrix}\right\}$ new file identifier↵

For example, to change the name of file LDATA to MDATA, the user types

-RENAME LDATA AS MDATA↵

The user may simply type RENAME followed by a Carriage Return, and the system prompts for the file identifiers. For example,

```
-RENAME↵
OLD NAME: PROG↵
NEW NAME: LPROG↵

-
```

The system prevents the user from changing the file identifier to an existing file identifier.

The user may change several file identifiers with one command by using *, #, or ? notation. For example, the command

-RENAME TEST1.* AS TEST6.*↵

changes the names of all files named TEST1, regardless of extension, to TEST6 with the same extension. The command

-RENAME EOM### AS ACCT###↵

renames all files whose names begin with EOM and contain six characters to ACCT followed by the last three characters of the previous name; that is, EOM122 is changed to ACCT122, EOMNTH is changed to ACCTNTH, etc.

PRINTING FILES

The TYPE, PRINT, and LIST commands allow the user to print files on the terminal or on the line printer at Tymshare's computer center.[1]

The TYPE command prints a file(s) on the terminal; the LIST command prints a file(s) on the line printer in page format with a heading on each page, and the PRINT command prints a file(s) on the line printer without page format and without a heading on each page. A heading consists of a page number, the file name, the date, and the time.

The switches described in the subsections below change the format of the listings produced by each of the printing commands.

1 - The user should consult his Tymshare representative if he wants to arrange to use a local line printer.

The Form of the Printing Commands

The general form of the printing commands is

-{TYPE / PRINT / LIST} file list↵

where the file list is one or more file identifiers and optional switches which control the form of the output.

Examples of valid print commands are listed below.

Command	Effect
TYPE FILE1	Prints the contents of file FILE1 on the terminal
PRINT DAT1,DAT2	Prints the contents of files DAT1 and DAT2 on the line printer with no headings[1]
LIST AA,BB,CC	Prints the contents of files AA, BB, and CC on the line printer.[1] Each file starts on a new page and each page begins with a heading
PRINT /SMALL FORTFL,A1	Prints the contents of file FORTFL and A1 on the line printer, 43 lines per page[2]

The user can store a copy of the printed output on a disk file by specifying a file identifier after the command. For example, the command

-LIST PRFIL=DAT1,DAT2↵

stores the contents of files DAT1 and DAT2 on PRFIL in page format with a heading on each page.

Specifying Format Switches

The TYPE, PRINT, and LIST commands allow the user to control the format of the listing by altering any of the preset switches. Switches may be set to apply to one or more files named in the command.

A switch applies to only one file if it immediately follows a file identifier and is separated from subsequent file identifiers by a comma (,).[3] For example, the command

-PRINT DATCOR/HEAD,GPFIN,PALALT↵

prints the listing of DATCOR with headings, but prints GPFIN and PALALT without headings.

1 - The line printer is located at Tymshare Computer Operations.
2 - The /SMALL switch is described on page 28.
3 - The /COUNT switch is an exception to this rule. See page 28 for a description of this switch.

To make a switch apply to several files, it must precede the list of file identifiers. For example, the command

```
-PRINT /HEAD FILEA,DATFIL,BDATA↵
```

prints each of the specified files with headings.

These two methods of specifying format switches may be combined in the same command. A switch that applies to only one file overrides a switch that applies to several files. For example, in the command

```
-PRINT /ONENUM /HEAD VOLE.BAL,HANBAL,BASE.BAL/NOHEAD,FOOT.BAL↵
```

the switch /NOHEAD cancels the switch /HEAD for the printing of BASE.BAL, but /HEAD is in effect for the file FOOT.BAL.

The following table lists both the preset and the optional switches for the TYPE, PRINT, and LIST commands. Each switch may be abbreviated to as few characters as necessary for uniqueness. For example, /DOUBLE may be abbreviated to /D, and /NOHEAD may be abbreviated to /NOH.

Control	Switch	Result	Preset For		
			LIST	PRINT	TYPE
Spacing	/SINGLE	Single spacing	•	•	•
	/DOUBLE	Double spacing			
	/MULTISPACE *n*	Spacing of *n* lines between printed lines			
Paper size	/LARGE	58 lines per page printed; standard for 11- by 14 7/8-inch paper	•	•	
	/SMALL	43 lines per page printed; standard for 8 1/2- by 11-inch paper			
	/SIZE *n*	Page length of *n* lines per page, including any heading and blank lines			• *n*=54
Line length	/LOL *n*	Line length of *n* characters; *n* must be an integer greater than or equal to 20			
Page count	/COUNT	The total number of pages in the listing printed at the user's terminal *NOTE: Once this switch is specified, all file names typed after it are affected by it*			

Control	Switch	Result	Preset For		
			LIST	PRINT	TYPE
Heading	/HEAD	Heading printed on each page of the listing. The heading consists of the page number, file name, date, and time of printing	•		
	/NOHEAD	Heading not printed		•	•
	/CHEAD	File identification information printed at the beginning of the file listing on a separate page. The place of printing, system number, directory name, file name, date, and time of printing are printed			
	/NOCHEAD	File identification information not printed	•	•	
	/TWONUM	Two-digit page numbers (PAGE 1-1, PAGE 1-2, ..., PAGE *m-n*) printed on each page of the listing. The first number is the logical page number and is incremented each time a top-of-form character, Control L, is encountered in the text of the file being printed. The second number is the sequential subpage number of the logical page number[1] *NOTE: Since the page number is part of the heading, the user must also enter the /HEAD switch in conjunction with either the /ONENUM or the /TWONUM switch if he uses the PRINT command*	•	•	
	/ONENUM	Pages numbered sequentially (PAGE 1, PAGE 2, ..., PAGE *n*)			
Case	/CASE	Both uppercase and lowercase alphabetic characters printed, as well as numbers, 27 special symbols, and the space	•	•	•
	/NOCASE	Same as /CASE, except lowercase alphabetic characters printed as uppercase alphabetic characters			
	/NOFULL	The special characters ({ } ` ~ | RUBOUT) not available	•	•	
	/FULLCH	Full set of 96 characters available			

(Table continues)

1 - A Control V must be used to insert a Control L as text in the file itself. For further information on the use of control characters, refer to the *Tymshare EDITOR Reference Manual.*

Control	Switch	Result	Preset For		
			LIST	PRINT	TYPE
Special case	/NOQUESTION	Control characters and control shift characters not printed	•	•	•
	/QUESTION	Control characters and control shift characters printed as ?"character". For example, S^C is printed as ?. and H^C is printed as ?(			
Special program	/SEQUENCE	Line numbers contained in a sequentially numbered file are printed	•	•	•
	/NOSEQUENCE	Line numbers contained in a sequentially numbered file are not printed			
	/FORTRAN	File printed as a FORTRAN data file if it contains FORTRAN carriage control as the first character in each line of the file			
	/NOFORTRAN	The effect of the /FORTRAN switch nullified	•	•	•
Default switches	/NORMAL	All preset switches restored			

Refer to the *Tymshare Remote Line Printer Software Reference Manual* for a description of all commands and switches that apply to remote line printers.

In the example below, the user prints the file TRIG on the terminal with double spacing and a heading.

```
-TYPE TRIG /DOUBLE /HEAD↵



PAGE 1-1          TRIG      MON 29-APR-74 17:40

        WRITE (5,10)

        READ (5,100) ANGLE

        X=SIN(ANGLE)

        Y=COS(ANGLE)

        WRITE(5,20)X,Y

10      FORMAT (14H ENTER ANGLE: )
```

```
20      FORMAT (6H SIN= ,F5.2,5X,6H COS= ,F5.2)

100     FORMAT(F7.2)

        END

-
```

COMPARING THE CONTENTS OF TWO FILES

The DIFFERENCES command compares the contents of two files specified by the user and lists all differences between the files. The general form of the DIFFERENCES command is

–DIFFERENCES [file identifier$_0$ =] file identifier$_1$, file identifier$_2$ [switch(es)]↵

If file identifier$_0$ = is omitted, the differences are listed on the terminal. The user may cause the system to prompt for the input and output file identifiers by typing the command name and optional switches followed by a Carriage Return. For example,

–DIFFERENCES [switches]↵
OUTPUT TO: file identifier$_0$↵
INPUT FILE 1: file identifier$_1$↵
INPUT FILE 2: file identifier$_2$↵

file identifier$_0$	Indicates the output file; it may be any disk file identifier or TER for the terminal.
file identifier$_1$ *and* file identifier$_2$	Specify the files to be compared.
switches	Represents any of the DIFFERENCES switches described below.

The DIFFERENCES command compares files in any one of three modes: symbolic, binary, or core image.[1] Files compared in the symbolic mode are compared line by line; files compared in the binary or core-image mode are compared word by word. Octal locations printed during core-image comparisons are core addresses rather than locations within the file. The comparison mode is determined by the switches entered with the DIFFERENCES command or, if no switch is specified, by file name extensions.[2] The extensions that imply a core-image comparison are HGH, LOW, SAV, and SHR. The following extensions imply a binary comparison:

BAC	**DCR**	**QUC**	**RTB**
BIN	**DMP**	**QUE**	**SFD**
BUG	**MFD**	**QUF**	**SYM**
CAL	**MSB**	**REL**	**SYS**
CHN	**OVL**	**RIM**	**UFD**
DAE	**OVR**	**RMT**	**XPN**

All other extensions imply a symbolic comparison.

1 - Symbolic files contain symbolic source code or text; binary files contain relocatable object code; core-image files contain an exact copy of a program and subprograms loaded in core.
2 - If the extension on file identifier$_2$ is not specified, it is assumed to be the same as the extension on file identifier$_1$.

When DIFFERENCES compares two symbolic files, it prints the file number and page number(s) for each file, followed by all lines that are not identical in both files; if the files are the same, DIFFERENCES prints NO DIFFERENCES ENCOUNTERED. For reference purposes, DIFFERENCES follows each set of nonidentical lines with a line that matches in both files. For example,

```
-DIFFERENCES↵
OUTPUT TO: TER↵
INPUT FILE 1: TRIG.F4↵
INPUT FILE 2: TRIG1.F4↵

1)1             WRITE (5,10)
1)              READ (5,100) ANGLE
1)              X=SIN(ANGLE)
****
2)1             WRITE (1,10)
2)              READ (0,100) ANGLE
2)              X=SIN(ANGLE)
**************

Files are different


-
```

DIFFERENCES prints the lines from TRIG.F4 that do not match the corresponding lines in TRIG1.F4.

This line is the same in both files.

DIFFERENCES prints the lines from TRIG1.F4 that do not match the corresponding lines in TRIG.F4.

This line is the same in both files.

When DIFFERENCES compares two binary files, it prints the nonidentical words from each file side by side. Each line of the comparison has the following format:

octal location first file word second file word difference[1]

For example, the binary relocatable files from the compilation of TRIG and TRIG1 are compared below.

```
-DIFFERENCES TRIG.REL,TRIG1.REL↵

000011  017040 000005   017040 000001          000004
000016  016040 000005   016040 000000          000005

Files are different


-
```

When DIFFERENCES compares two core-image files, it prints the same comparison as for binary files, but the octal location is a core address rather than a location in the file.

1 - If only the right halves of the words differ, the difference printed represents the subtraction of one word from the other. If the left halves of the words differ, the difference printed represents the exclusive OR of the two words; that is, a bit-by-bit comparison results in true if one or the other is true, but not if both are true or both are false.

The user can include switches in the DIFFERENCES command to specify a particular type of comparison. If a switch and a file name extension conflict, the switch overrides the extension. The switches available for symbolic and binary comparisons are listed below, as well as the function of each switch.

Switch	Function
/ASCII (*or* /SYMBOLIC)	Compares in ASCII mode
/BLANK	Compares blank lines. If this switch is not used, blank lines are ignored
/COMMENT	Ignores comments, that is, all text on a line after a semicolon (;). Spacing is also ignored
/SPACING	Ignores spacing and tabs. The following two lines match if this switch is on: **AB CD EF** **ABC DE F**
/HELP	Prints the list of available switches
/LOWER *n*	Sets *n* as the first word to be compared in a partial binary comparison; *n* is an octal number
/UPPER *n*	Sets *n* as the last word to be compared in a partial binary comparison; *n* is an octal number
/QUICK	Compares files and notifies the user whether or not they match; does not print nonidentical lines
/UPDATE	Causes the second file to be written on the output file with flags next to the lines that differ from the first file (update mode)
/WORD	Compares actual file contents without core-image expansion
/EXPAND	Expands core-image files before comparing them

Section 4
THE USER FILE DIRECTORY

The user file directory (UFD) contains descriptive information about each file in the user's storage area. Tymshare's file security controls provide maximum security and determine who can access the directory and the files, who can read them, and who can write on them.

XEXEC has commands to set security controls on individual files and on an entire file directory. This section discusses how to set the controls using the DECLARE and FDC commands and how to use switches to list information about all the files or specific files.

FILE SECURITY CONTROLS

The user sets controls on individual files with the DECLARE command, and on the entire file directory with the FDC (File Directory Controls) command. If there is a conflict between these controls, the more restrictive protection takes precedence. For example, if a user protects his entire directory from the public's sharing his files, an individual file is not sharable, regardless of the individual controls set for it with the DECLARE command.

Declaring and Printing General Protection for the File Directory

The user sets controls on the entire file directory for two classes of users: ACCOUNT and PUBLIC. The ACCOUNT controls refer to controls on the directory with respect to other users in the same account;[1] the PUBLIC controls refer to controls on the directory for all other users on the computer system.

To set the controls on his entire file directory, the user types

-FDC↵

The system responds with the same three questions for each of the two classes of users:

Question	Y Response Enables Others To
SHARABLE?	Access the user's files
NEW FILES?	Add new files to the user's directory
LISTABLE?	List file directory information. This does not permit others to list the contents of the user's files

1 - The term ACCOUNT refers to all users having the same Account Supervisor.

The user responds to each question with a Y or N, followed by a Carriage Return. For example,

```
-FDC↵
ALLOW ACCOUNT:
  SHARABLE? Y↵
  NEW FILES? Y↵
  LISTABLE? Y↵
PUBLIC:
  SHARABLE? Y↵
  NEW FILES? N↵
  LISTABLE? N↵

-
```

The PFDC (Print File Directory Controls) command prints the controls established for the user's file directory. For example,

```
-PFDC↵

ACCOUNT: SHARABLE NEW FILES LISTABLE
PUBLIC: SHARABLE
-
```

If the user has typed N in response to all questions in the FDC command, that is, if the user sets all restrictions, PFDC does not print anything.

The user need only type the FDC command to alter the standard protection. Tymshare currently sets the initial file directory controls for new users as follows:[1]

ACCOUNT: SHARABLE LISTABLE
PUBLIC: SHARABLE

When using the FDC command, the user may type a dash (-) in response to the first question in either set to request the standard protection for that class of users.[2] In the following example, the user restores the directory controls to the standard protection.

```
-FDC↵
ALLOW ACCOUNT:
  SHARABLE? -↵
PUBLIC:
  SHARABLE? -↵

-PFDC↵

ACCOUNT: SHARABLE LISTABLE        This is the standard protection.
PUBLIC: SHARABLE
-
```

1 - An individual file becomes sharable only when so declared. See page 37 for the discussion of the DECLARE command.
2 - The dash option also works with the prompting form of the DECLARE command shown on page 39.

Declaring File Security Controls

The DECLARE command sets file security controls for individual files. These controls recognize three classes of users: PRIVATE, ACCOUNT, and PUBLIC. For each class, the user may control the extent of access to be permitted.

The general form of the DECLARE command is

-DECLARE private account public file list↵

private	Specifies the file security controls for the user himself.
account	Specifies the file security controls for other users in the same account.
public	Specifies the file security controls for users on the Tymshare system who are not in the same account.
file list	Indicates the file or files for which the controls are established. Multiple file names are separated by spaces or commas. Switches may be used to select files according to certain criteria. These switches are described on page 42.

The file security controls that can be entered for PRIVATE, ACCOUNT, and PUBLIC users are shown in the table below. The table is arranged in order of controls allowing the most privileges to controls allowing the least privileges so that a given file security control implies all the privileges of the controls following it in the table.

FILE SECURITY CONTROLS

Symbol	Protection
ALL	Allows deletion of the file
CP	Allows change of protection of the file
UPD	Allows changes and additions to the file
RD	Allows reading of the file
RUN	Allows executing the file
LK	Allows access to the directory information pertaining to the file
NO	Allows no access to the file

For example, the command

-DECLARE ALL RUN NO PROG1, PROG2↵

establishes the following protection for files PROG1 and PROG2. The user himself can access, modify, or delete the files as he chooses. Other users in the same account can execute the files and obtain directory information about them. Users on the system who are not in the same account have no access to or knowledge of the files.

The system prompts the XEXEC user if he types

-DECLARE↵

by asking a series of questions for each user class that can be answered by a Y or N followed by a Carriage Return. The system prints

```
ALLOW PRIVATE:
    MODIFY?
```

If the user answers N, no modifications are allowed and the system skips the next three questions. If the user answers Y, the system prints

RENAME AND DELETE(ALL)?
CHANGE PROT(CP)?
UPDATE(UPD)?

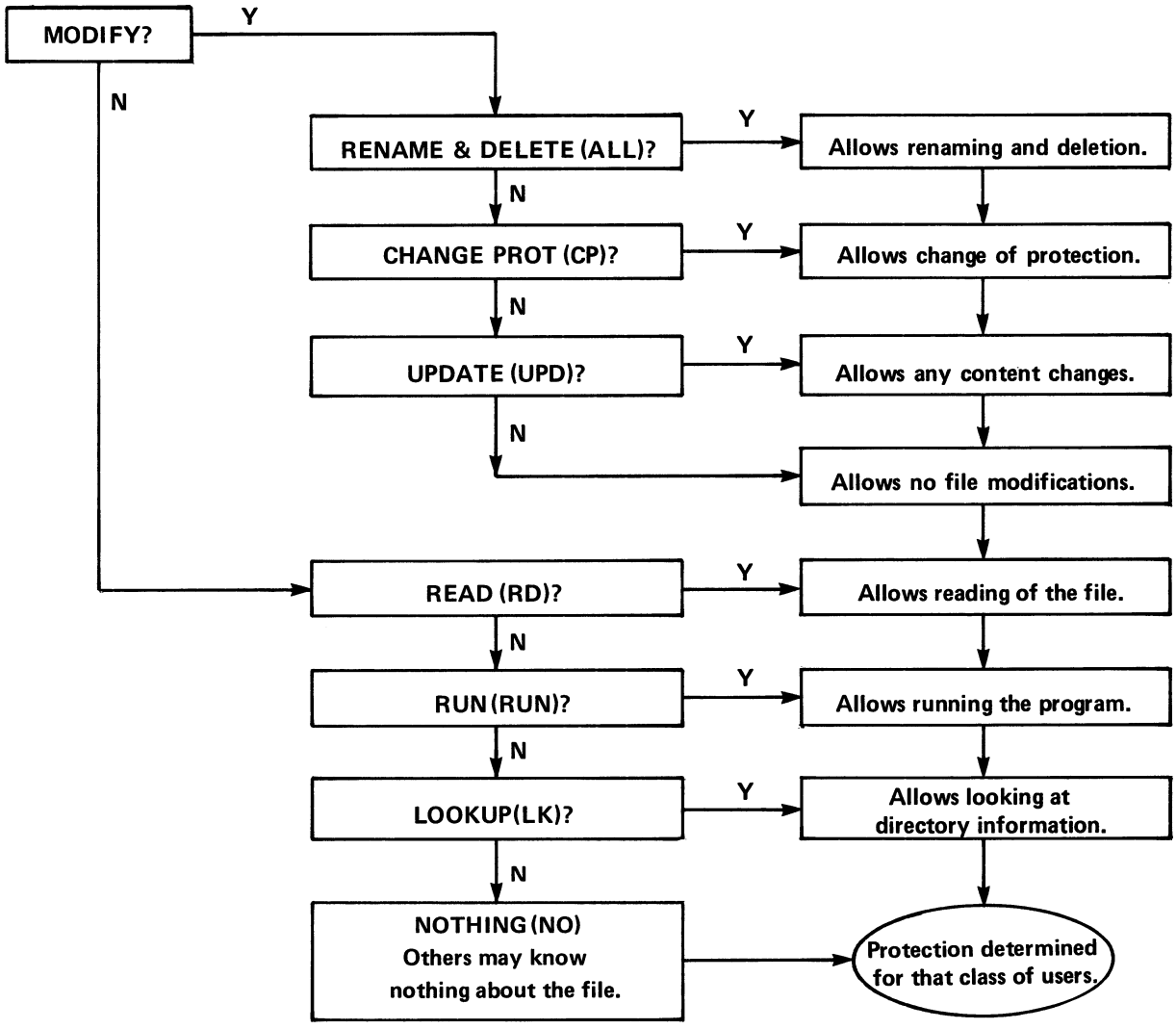
If the user responds to any of these questions with a Y, the remaining questions are skipped. If the user answers N, the system prints

READ(RD)?
RUN(RUN)?
LOOKUP(LK)?

If the user responds to any of these questions with a Y, the remaining questions are skipped.

The same sequence of questions is then repeated for ACCOUNT, or users in the same account, and for PUBLIC, or users on the system who are not in the same account.

The following chart details the specific questions asked and the pattern for all classes of users: PRIVATE, ACCOUNT, and PUBLIC.



After the user determines the protection for PRIVATE, ACCOUNT, and PUBLIC, the system prints

FILE(S)?

and the user types file names and/or switches to specify the files that are to receive this protection. The switches that may be used to select files are described on page 42.

In the example shown below, the user wants to change the protection on a group of his files to allow himself to do anything to the files, to allow other users in his account to run but not to read them, and to prevent other Tymshare users from accessing the files in any way.

```
-DECLARE↵
ALLOW PRIVATE:
  MODIFY? Y↵
  RENAME & DELETE(ALL)? Y↵
ACCOUNT:
  MODIFY? N↵
  READ(RD)? N↵
  RUN(RUN)? Y↵
PUBLIC:
  MODIFY? N↵
  READ(RD)? N↵
  RUN(RUN)? N↵
  LOOKUP(LK)? N↵
  NOTHING(NO)
FILE(S): PGM1.*,PGM2.*↵

FILES RENAMED:
PGM2     SAV
PGM2     LST
PGM1     CRF

-FILES /PROTECTION PGM1.*,PGM2.*↵
```

The FILES command is discussed on page 41.

```
PGM2     SAV   ALL RUN NO
PGM2     LST   ALL RUN NO
PGM1     CRF   ALL RUN NO

-
```

NOTE: The user can always change the protection for his own files even if the private protection is NO.

Tymshare initially sets the controls for each file as

ALL RD NO

meaning that the user can do anything to his files, other users in his account may only read or run them, and users outside his account may not access or utilize them in any way. Therefore, the user types the DECLARE command only to alter the standard protection. He may type a dash (-) in response to the first question to request the standard protection for that class of users.

For example, the user can set the same file controls used in the previous example more quickly by using the dash.

```
-DECLARE↵
ALLOW PRIVATE:
  MODIFY? -↵
ACCOUNT:
  MODIFY? N↵
  READ(RD)? N↵
  RUN(RUN)? Y↵
PUBLIC:
  MODIFY? -↵
FILE(S): FCALC↵

-FILES /PROTECTION FCALC↵

FCALC          ALL RUN NO

-
```

Declaring Files Accessible Through Program Use

The SETLIC system program allows a user to set a home files license. This license allows others to access to the user's files which are utilized by a sharable file in his directory. For example, user ALLEN has a program on the file BUSTAT that he has given RUN protection. This program uses the following files in his directory: ACCT, INV, and DATBS. He uses the SETLIC program to allow other users access to ACCT, INV, and DATBS when they run BUSTAT. He executes SETLIC as follows:

```
-R SETLIC↵

FILE NAME: BUSTAT↵
HOME FILE ACCESS? Y↵

EXIT

-
```

LISTING FILE INFORMATION

XEXEC offers commands which print descriptive information about the files in the user's directory. The FILES command prints the names and extensions of the files in the directory. The DIRECTORY command prints the names and extensions of the files as well as the size, creation date and time, date of most recent access, and protection of each file.

The user can request that the FILES or DIRECTORY command print information about specified files instead of the entire directory. For example, the command

-DIRECTORY PDATA, PGM.SAV, TDATA↵

prints the directory information for the files PDATA, PGM.SAV, and TDATA only.

When referring to specific files in the directory, the user may use the special notation *, #, ?, NOT (or –) and ALL to specify the desired files.[1] For example, if the user wants to list all his files except those with the REL extension, he uses the command

–FILES ALL–*.REL↵

The user can modify the information printed by the FILES or DIRECTORY command by including switches in the command. These switches may be used to request additional information about files, to specify the order of listing files, or to select files according to various criteria. The switches are described on page 42.

The FILES Command

The FILES command prints the names and extensions of the specified files in the user's directory and any additional information requested by switches. The form of the command is

–FILES [file list]↵

where file list is a list of file names separated by commas, and optional switches.[2] If the file list is omitted, all files are listed.

Some examples of the FILES command are shown below.

Command	Result
FILES	Lists the names and extensions of all the user's files
FILES *.REL	Lists the names and extensions of the files with the extension REL
FILES /HELP	Prints all switches available with the FILES command
FILES /ALPHA /ACCESS	Lists all the user's files in alphabetical order, showing the most recent access date for each file

The DIRECTORY Command

The DIRECTORY command prints the name, extension, size, creation date and time, file protection, and date of most recent access for the specified files in the user's directory. The form of the command is

–DIRECTORY [file list]↵

where file list is a list of file names separated by commas, and optional switches.[2] If the file list is omitted, the command prints directory information for all files.

Some examples of the DIRECTORY command are shown below.

Command	Result
DIRECTORY	Prints the directory information for all the user's files
DIRECTORY PROG.*	Prints the directory information for all files with the name PROG regardless of extension
DIRECTORY /FAST	Prints the name and extension for all files

1 - The use of special notation when referring to files in commands is described on page 15.
2 - All switches are described on pages 42 and 43.

Switches

There are three types of switches: switches to request additional or restricted information about files, switches to specify the order of listing files, and switches to select files according to various criteria. A switch can be abbreviated to the minimum number of characters required to uniquely identify it and can be terminated by any nonalphanumeric character such as a blank or comma. The switches described in this subsection can be used with any of the following commands: FILES, DIRECTORY, DELETE, RENAME, and DECLARE.

The following switches request additional or restricted information about the user's files.

Switch	Information Requested
/ACCESS	Date of most recent access
/CREATION	Creation date
/FAST	File name and extension
/PROTECTION	Protection assigned to the file
/SECONDS	Creation date and time to the second
/SIZE	Size of the file (number of storage blocks of 640 characters)
/WORDS	Size of the file in words
/TIME	Creation date and time to the minute
/TOTAL	Number of storage blocks and files in the file list. This switch is automatic for the DIRECTORY or the FILES command with no file list
/EVERYTHING	Everything above, plus the mode of the file[1]

NOTE: The user may include several switches in a single command.

Files are normally listed in reverse chronological order by creation date; that is, the system prints the most recently created file first. If the user wants to see the files and corresponding information in a different order, the following switches are available.

Switch	Order Listed
/ALPHABETICAL	Alphabetically by file name and secondarily by extension
/EXTENSION	Files with extensions are arranged in alphabetical order by extension, followed by the files with no extensions
/STORAGE	Decreasing order of storage size; that is, the name of the largest file is listed first
/UNSORTED	Order in which files appear in the user's file directory
/REVERSE	Chronological order of creation. /REVERSE can be used with any other switch in this table to reverse the order established by that switch

1 - Mode information is of use only in machine language programming. See the *DECsystem10 Assembly Language Handbook* for a description.

NOTE: The switches in the preceding table must be used only one to a command except when the /REVERSE switch appears with another switch to reverse its order.

The remaining switches select files according to the criteria specified with the switch. All files in the user's directory except for those with the extension TMP are selected when a command is not followed by file designations or by any of the switches shown below.

Switch	Files Selected
/AFTER date	Files created on and after the date specified
/BEFORE date	Files created before the date specified
/TODAY	Files created during the present day
/TEMPS	All files including those with the extension TMP, which the system normally does not print

NOTE: The /AFTER and /BEFORE switches can be used together to indicate a range of dates.

The user specifies the date by the day, first three letters of the month, and last two digits of the year, all separated by dashes. For example,

29-SEP-73 21-NOV-73 5-JUN-74

If the user omits the year and/or month, the system assumes the current one. Therefore, if a user is working on the system on October 12, 1974 and wants to specify the date October 5, 1974, any of the three following ways of entering the date is acceptable:

/BEFORE 5-OCT-74

/BEFORE 5-OCT

/BEFORE 5

In the following example, the user requests a complete directory listing in alphabetical order, including time of creation to the second, for all files created on or after April 20.

```
-DIRECTORY /ALPH/SEC/AFTER 20-APR↵

BOM              5  23-APR-74 1350.54 ALL RD  NO   26-MAR-74(A)
BUSTAT           5  29-APR-74 1739.43 ALL RUN NO   30-APR-74(A)
INVEN            5  23-APR-74 1344.19 ALL RD  NO   23-APR-74(A)
PGM1    CRF      5  29-APR-74 1213.20 ALL RD  RD   30-APR-74(A)
PGM2    LST      5  29-APR-74 1751.57 ALL RUN NO   30-APR-74(A)
PGM2    SAV      5  29-APR-74 1752.41 ALL RUN NO   30-APR-74(A)
PO               5  23-APR-74 1350.53 ALL RD  NO   13-MAR-74(A)
TEXT    LST     10  24-APR-74 1438.35 ALL RD  NO   24-APR-74(A)
TEXT             5  24-APR-74 1438.26 ALL RD  NO   24-APR-74(A)
TRIG    F4       5  29-APR-74 1743.20 ALL RD  NO   30-APR-74(A)
TRIG    REL      5  29-APR-74 1745.10 ALL RD  NO   30-APR-74(A)
-
```

Size in blocks of 640 characters — *Creation date* — *Creation time* — *Seconds* — *File protection* — *Date of most recent access*

ACCESSING THE FILE DIRECTORY OF ANOTHER USER

The GFD (Get File Directory) command allows one user to gain access to another user's storage area, or directory. Access to a directory is granted only if the directory has been made sharable by its owner.[1] The form of the GFD command is

-GFD user name↵

When a user gains access to another user's directory using this command, he may perform any operations as though he had logged into that user name.[2] Once a user has gained access to another user's directory, he no longer has access to his own directory; he can access his own directory again by entering the GFD command with his own user name.

The GFD command allows more than one user to access the same file directory simultaneously. Thus, a set of files that is used very often by the member of an account can be stored in a sharable directory where they can be accessed by all the account's members. This reduces the storage costs for the account.

1 - See your Tymshare representative if you want to make your directory sharable.
2 - The GFD command also notifies the user of any damaged files in the directory.

Section 5
EDITOR, CONVERSATIONAL LANGUAGES, AND APPLICATIONS PROGRAMS

The user can access all TYMCOM-X languages and applications packages, as well as the EDITOR described on page 17, through XEXEC. The TYMCOM-X system offers its users a variety of conversational languages and a large, constantly increasing selection of applications programs.

The subsections which follow describe various ways of entering and leaving EDITOR, using conversational languages, and accessing applications programs. For a complete description of EDITOR, a particular language, or an applications program, refer to the appropriate Tymshare document.

ENTERING AND LEAVING THE EDITOR

This subsection describes the commands that transfer control between EDITOR and XEXEC.

Entering the EDITOR

The commands EDITOR, CREATE, and MODIFY access the EDITOR from XEXEC. The EDITOR command simply transfers control to the EDITOR. Its form is

-EDITOR↵

The CREATE command transfers control to EDITOR for the purpose of creating a file. Its form is

-CREATE file identifier↵

where file identifier specifies a file that does not yet exist.

The MODIFY command transfers control to EDITOR for the purpose of modifying an existing file. Its form is

-MODIFY [file identifier]↵

where file identifier refers to an existing file. If the file identifier is omitted, the command reads the file specified by the most recent CREATE or MODIFY command.

Leaving the EDITOR

The EDITOR commands QUIT, EXIT, and GO return control to XEXEC from the EDITOR.

The QUIT command simply returns control to XEXEC. If the user has added to or modified text in the EDITOR text area since entering EDITOR, the system prints

FILE NOT WRITTEN,OK?

and control returns to XEXEC only if the user types Y.

The EXIT command writes the current text on a file before returning control to XEXEC. The form of the command is

*EXIT [file identifier]↵

where file identifier specifies the file on which the EDITOR text is to be written. If a file identifier is not specified in the command, the output file is determined by the command used to enter the EDITOR. If the EDITOR was entered using the EDIT command, EDITOR prompts for a file name after the EXIT command. If the EDITOR was entered using the CREATE command, the current text is written on the file specified in the CREATE command. If the EDITOR was entered using the MODIFY command, the current text is written on the file specified in the MODIFY command; or, if none was specified, the text is written on the file specified in the most recent CREATE or MODIFY command.

The GO command performs the same functions as the EXIT command and then performs the most recent compile-type command specified before EDITOR was entered.[1]

The example shown below demonstrates a typical interaction between XEXEC and EDITOR.

```
-COMPILE PROG1↵                                    The user compiles PROG1.
F-IV:  PROG1
**                        10       FORMAT (15H ENTER ANGLE: )
**                                                      ^
**                S-3 ILLEGAL FIELD SPECIFICATION   The compiler
 MAIN.  ERRORS DETECTED: 1                          detects one
                                                    error.
? TOTAL ERRORS DETECTED: 1

EXIT

-MODIFY PROG1↵                                     The user calls EDITOR to correct PROG1.
240 CHRS
*:10:MODIFY↵
10       FORMAT (14H ENTER ANGLE: )↵               Line 10 is changed using EDITOR
*GO↵              The GO command writes the revised text   commands and control characters.
PROG1             on PROG1 and recompiles the program.
240 CHRS
F-IV:  PROG1
 MAIN.

EXIT

-
```

When the user interrupts an operation currently in progress in EDITOR by typing one or more Alt Mode/Escapes, the system returns control to XEXEC. The user may type any of three commands to return control to the subsystem: CONTINUE, REENTER, or START. The CONTINUE command returns control to EDITOR and continues the operation from the point of interruption. The REENTER command returns control to EDITOR command level and preserves the user's work as it was when he typed the Alt Mode/Escapes. The START command returns control to EDITOR command level and clears the user's work from the subsystem.

NOTE: The user may type any XEXEC command that does not cause the EDITOR text area to be lost before returning to his previous work in EDITOR.[2] *After entering any other XEXEC command, he may not return to EDITOR using the commands described above.*

1 - The compile-type commands are described on page 51.
2 - The commands that do not destroy the current core image are indicated in Appendix A, page 85.

CONVERSATIONAL LANGUAGES

Using the conversational languages available on the TYMCOM-X system, the user can develop, execute, and revise programs in an interactive environment. The user calls a language by typing its name after the XEXEC dash. For example,

-XBASIC↵

The user may then enter either program statements or commands in response to a prompt character. The prompt character printed depends upon which language is being used.

APPLICATIONS PROGRAMS

The TYMCOM-X has a large library of applications programs. The emphasis in these programs is on conversational problem solving. The applications programs include data management systems, linear programming packages, statistical aids, engineering programs, computer simulation packages, financial forecasting systems, and many others.

The XEXEC user calls most applications programs by typing

-R program name↵

For example,

-R STATPAK↵

Some applications programs are part of the User Program Library. These programs are called by typing

-RUN (UPL)program name↵

For example,

-RUN (UPL)MACE↵

To access a particular applications program, the user should refer to the appropriate reference manual or consult his Tymshare representative.

Section 6
RUNNING USER PROGRAMS IN XEXEC

This section explains the methods for executing a previously created program. First, the system must read and translate the source program into internal machine language; this is the compilation phase. Next, the system must organize and move the program's relocatable code and all subprograms used by the program into core memory; the name of this step is *loading*. Finally, the system executes the loaded instructions. If errors are encountered during execution, the user can use a debugging package to examine the execution of his program. The commands that perform these steps are presented in this subsection.

The following table summarizes all the functions performed by each command used in program execution.

Function / Command	Compile	Load	Debug	Execute	Other
COMPILE	•				
LOAD	•	•			
EXECUTE	•	•		•	
CDEBUG	•	•	Loads COBOL debugger with the program		
FDEBUG	•	•	Loads FORTRAN debugger with the program		
DEBUG	•	•	Loads DDT with the program		
TRY	•	•	Loads DDT with the program	•	
DDT			Enters DDT environment if DDT is loaded		
D					Deposits information in a core location
E					Examines specified core location

(Table continues)

Function / Command	Compile	Load	Debug	Execute	Other
START				•	
CSTART				•	Allows simultaneous execution of any command that does not destroy current core image[1]
CROSS					Produces cross-reference listing of files compiled in the current session, using the CREF switch
CONTINUE				Continues execution from point of interruption	
CCONTINUE				Continues execution from point of interruption	Allows simultaneous execution of any command that does not destroy the current core image[1]
SAVE					Saves core image on file
SSAVE					Saves core image on file. High segment, if present, is to be sharable
GET		Loads file created by SAVE or SSAVE			
GO RUN		Loads file created by SAVE or SSAVE		•	

1 - The commands that do not destroy the current core image are indicated in Appendix A, page 85.

Function / Command	Compile	Load	Debug	Execute	Other
FINISH					Terminates any input or output currently in progress on the specified device and closes the specified device
CLOSE					Writes end-of-file indicator and closes the specified file

THE COMPILE-TYPE COMMANDS

The commands COMPILE, LOAD, EXECUTE, CDEBUG, FDEBUG, DEBUG, and TRY are called compile-type commands because each of these commands includes compilation as one of its functions. The compile-type commands compile a program only if a compiled version of the current source code does not exist. The form of a compile-type command is

-command name [file list]↵

command name — COMPILE, LOAD, EXECUTE, CDEBUG, FDEBUG, DEBUG, or TRY.

file list — Indicates a file identifier or a list of file identifiers. Multiple file identifiers are separated by commas. It may also include any of the switches described on page 55. If the file list is omitted, the command uses the file(s) indicated in the most recent compile-type command.

NOTE: The user may not use the asterisk (*), crosshatch (#), question mark (?), NOT (or –), or ALL notation to refer to files in this command; he must specify each file individually.

The compile-type commands produce relocatable binary files of the specified programs and list diagnostic messages on the terminal. The relocatable binary file of a program has the same name as the program and the extension REL. For example, the command

-COMPILE MAIN↵

produces a file named MAIN.REL. No compilation occurs when a relocatable file newer than the source file already exists for the specified file.[1] In other words, the system does not recompile a program unless the source program file has been changed since the last compilation. For example,

```
-COMPILE PGM5↵
F-IV:  PGM5                    The compiler message indicates the language translator used.
 MAIN.

EXIT

-COMPILE PGM5↵                 The system does not recompile PGM5.

-
```

1 - See page 57 for a discussion of the forced-compile switch to alter the usual procedure.

The extension of the source file identifier determines which processor converts the source program into internal machine language. The user need not specify the extension in the command unless it is not a standard language extension. The system checks for the standard extensions and compiles with the appropriate processor according to the table on page 14.

NOTE: When the file name has a nonstandard extension or no extension, the system uses the FORTRAN IV compiler.

Each time the user types a COMPILE, LOAD, EXECUTE, CDEBUG, FDEBUG, DEBUG, or TRY command with a file list, the system stores the command and the file list. The next time the user types a compile-type command, he can omit the file list and the system will use the file list entered with the previous compile-type command. For example, the user enters the file list, /COMPILE FCALC, with the COMPILE command.

```
-COMPILE /COMPILE FCALC↵
F-IV:  FCALC
 MAIN.
 FACTRL

EXIT

-
```

He then enters a LOAD command without a file list and the system uses the file list, /COMPILE FCALC.

```
-LOAD↵
F-IV:  FCALC
 MAIN.
 FACTRL
LOADING
5K CORE

-
```

The system recompiles the program because the stored file list contains the /COMPILE switch.

NOTE: When the user takes advantage of the stored file list, he can only type the command word, such as COMPILE, LOAD, EXECUTE, CDEBUG, FDEBUG, DEBUG, or TRY; he cannot add to the file list.

If the user wants to work with another file or if the next command does not apply to all files and switches specified in the previous command, the user must respecify the file name(s) and/or switches.

The COMPILE Command

The COMPILE command translates each source program specified in the file list to its machine language equivalent producing a relocatable binary file and any appropriate error messages. For example, the command

```
-COMPILE ABC, COMP.CBL, XFILE.F4↵
```

translates ABC and XFILE with the FORTRAN IV compiler and COMP with the COBOL compiler, since F4 and CBL are standard extensions and FORTRAN IV is the default processor. The extensions could have been omitted since they are standard.

The LOAD Command

The LOAD command compiles, if necessary, and then loads the programs and subprograms stored on each file in the file list. The LOAD command also attempts to LOAD all subprograms called by the main program (the first program loaded). The user must include in the LOAD command the names of the files that contain the required subprograms unless these subprograms are provided by the system. For example, the command

```
-LOAD PGM5↵
```

compiles PGM5, if necessary, loads the file PGM5.REL, and attempts to load any subprograms called by PGM5. Suppose PGM5 calls subroutine SUBR, which is not a system subprogram. The subroutine SUBR must be included on file PGM5 or the user must load with the program a file that contains the subroutine. For example,

```
-LOAD PGM5, SUBFIL↵
```

where SUBFIL contains the subprogram SUBR.

The EXECUTE Command

The EXECUTE command compiles each file in the file list, if necessary, loads the compiled program and associated subprograms into core, and begins execution. In the example shown below, the user compiles, loads, and executes the program TRPRG and the subroutine SINCOS, which is called by TRPRG.

```
-EXECUTE TRPRG, SINCOS↵
F-IV:  TRPRG
 MAIN.
F-IV:  SINCOS.F4
 SINCOS
LOADING
EXECUTION

ENTER ANGLE:
1.5↵

SIN=  1.00       COS=  0.07
EXIT

-
```

The CDEBUG Command

The CDEBUG command compiles each file in the file list, if necessary, loads the compiled program and associated subprograms into core, loads the COBOL debugger, and begins execution.[1] At this point, the user may set breakpoints, execute his program one statement at a time, or perform any of the CDEBUG operations.

NOTE: The relocatable files used by the COBOL debugger must be created by the CDEBUG command.

In the example below, the user compiles and loads the program MINPGM, then enters the COBOL debugger, sets a breakpoint, and begins execution.

```
-CDEBUG MINPGM↵                          The CDEBUG command compiles, loads, and
COBOL:  MINPGM  [MINPGM.CBL]             executes MINPGM.
LOADING
EXECUTION

STARTING COBOL DDT                       The COBOL debugger is entered.

*BREAK TOT-LOOP↵                         The user sets a breakpoint at TOT-LOOP.

*PROCEED↵                                The user begins execution of the program.
ADDING MACHINE-
BREAK AT  <<TOT-LOOP>>                   The breakpoint at TOT-LOOP is encountered.
*
```

The FDEBUG Command

The FDEBUG command compiles each file in the file list, if necessary, loads the compiled program and associated subprograms into core, and enters the FORTRAN debugger.[2] At this point, the user may perform any of the FDEBUG operations.

NOTE: The relocatable files used by the FORTRAN debugger must be created by the FDEBUG command.

In the example shown below, the user calls FDEBUG with the file TRIG, which has already been compiled by a previous FDEBUG command. The command loads TRIG and enters the FORTRAN debugger; the user sets the value of variable X to 5.

```
-FDEBUG TRIG↵
LOADING

DEBUG:
*SET X=5↵

*
```

1 - For more information about the COBOL debugger, refer to the discussion of COBDDT in the *Tymshare TYMCOM-X COBOL Reference Manual.*

2 - The FORTRAN debugger is described in the *Tymshare TYMCOM-X DEBUG Reference Manual.*

The DEBUG Command

The DEBUG command can be used to debug MACRO programs or FORTRAN programs. It compiles each file in the file list, if necessary, loads the compiled programs and associated subprograms into core, and enters the DDT debugger.[1]

In the example shown below, the user compiles and loads the MACRO assembly language program MULT.MAC and enters DDT using the DEBUG command. He then sets a breakpoint and begins execution.

```
-DEBUG MULT↵
MACRO: MULT
LOADING
DDT

NLOOP+2/    MOVEM V,BUNAP(ME)    .;B    ;G

;;1B>NLOOP+2
```

The TRY Command

The TRY command begins execution of a program with all the debugging tools available for use. It is identical to the DEBUG command except that it begins execution of the program indicated by the file list. In the example shown below, the user specifies the same file with the TRY command as with the DEBUG command in the previous example. The command does not compile MULT.MAC because a relocatable file already exists for the file MULT. The command loads the program and begins execution.

```
-TRY MULT↵
LOADING
EXECUTION
```

Command Switches

The user can modify the COMPILE, LOAD, EXECUTE, CDEBUG, FDEBUG, DEBUG, and TRY commands with a variety of switches. The user precedes the switch with a slash (/) and terminates the switch with any nonalphanumeric character, usually a space or a comma (,). The user may abbreviate the switch name, provided that he uniquely identifies it.

A switch may apply either to a single file or to many files in a command. A single-file switch immediately follows a file identifier and applies only to that file. For example,

```
-COMPILE X, Y/MACRO, Z↵
```

The /MACRO switch applies only to file Y.

1 - The DDT debugger is described in the *DECsystem10 Assembly Language Handbook*.

A multifile switch applies to all files following the switch unless the user cancels the switch with a subsequent switch. In the examples given below, the /LIST switch applies to files A and B only.

```
-COMPILE /LIST A,B↵
-COMPILE D, /LIST A,B↵
-COMPILE D, /LIST A,C/NOLIST,B↵
-COMPILE D, /LIST A,B,/NOLIST C,Z,V↵
```

Processor Switches

Processor switches allow the user to change the default processor for files listed in a compile-type command or to specify the processor for a particular file. The processor switches are listed in the table below.

Switch	Processor
/COBOL *or* /CO	COBOL
/MACRO *or* /M	MACRO
/FORTRAN *or* /F	FORTRAN IV

Normally, FORTRAN is the default processor for files without standard extensions listed in a compile-type command. Thus, in a compile-type command that does not have processor switches the FORTRAN processor compiles all files without standard extensions and the processors indicated by the extensions compile files with standard extensions.[1] To change the default processor for a list of files, the user inserts the processor switch in the file list before the file name(s) he wants to affect. For example, in the command

```
-COMPILE AA, /MACRO BB, CC↵
```

the /MACRO switch applies to file names BB and CC. When a processor switch is used this way, it is overridden by a standard extension in a file identifier. For example, if BB in the command shown above has the extension F4 and CC has no extension, then BB is compiled by FORTRAN and CC is compiled by MACRO.

A processor switch applies to a single file if it immediately follows the file name in a command. For example, in the command

```
-COMPILE AA, BB/MACRO, CC↵
```

the /MACRO switch applies to file name BB only. A processor switch used this way overrides the extension of the file identifier; that is, BB in the example shown above is compiled by MACRO even if it has an extension such as F4 or CBL.

Listing Switches

Switches are available to generate ordinary listings and cross-reference listings of the file. The operation of the switch produces a file with the same name and the extension LST or CRF, as appropriate.

1 - A list of standard XEXEC extensions and their meanings is shown on page 14.

The /LIST switch generates a listing of the file and a list of the subprograms referred to in the main program. The /NOLIST switch is available to cancel the /LIST switch.

For example, when the command

```
-COMPILE /LIST A,B/NOLIST,C↵
```

is executed, it creates the files A.REL, A.LST, B.REL, C.REL, and C.LST.

The /CREF (or /CROSS) switch is the same as the /LIST switch except that it produces a cross-reference listing, which the user accesses with the CROSS command.[1]

Compilation Control Switches

Compilation normally occurs when no relocatable file exists for the source file or when the source file is newer than the existing relocatable file. If the relocatable file is newer than the source file, the system normally does not perform the compilation.

However, there are situations when the user may want an extra compilation, for example, when he desires a listing. To force the compilation, he uses the /COMPILE switch. For example, the command

```
-COMPILE /CREF /COMPILE A↵
```

creates the cross-reference listing file A.CRF, although a current A.REL file already exists. The system also re-creates the relocatable file.

The /NOCOMPILE switch cancels the effect of the /COMPILE switch.

The /REL switch causes a compile-type command to use the existing relocatable file even if a newer source file exists.

Library Search Switch

Normally, when a program is loaded with a file that contains associated subprograms, the whole file is loaded into core. The /LIBRARY switch allows the user to load only selected subroutines from a file that may contain many subroutines. This can save a significant amount of core space. If, for example, the user's program calls three subroutines that are written on a file containing twenty subroutines, he can use the /LIBRARY switch to select only those subroutines used by his program.

In the example shown below, the main program is on the file MAINPG. The main program calls subroutines SUB1, SUB2, and SUB3. Subroutine SUB2 is stored on file MAINPG after the main program. Subroutines SUB1 and SUB3 are stored on the file SUBODD, along with several other subroutines. To load only subroutines SUB1 and SUB3 from SUBODD, the user executes the program as follows:

```
-EXECUTE MAINPG, SUBODD/LIBRARY↵
```

If the /LIBRARY switch were not included in the command, all the subroutines stored on the file SUBODD would be loaded into core.

NOTE: The system automatically searches the FORTRAN library to locate standard library routines.

1 - The CROSS command is described on page 61.

Compiler and Assembler Switches

Compiler and assembler switches request options from the compiler or assembler for a specific language. The switch consists of a letter, or letters, enclosed in parentheses; it must immediately follow a file name. For example, the FORTRAN switch option to suppress output of error messages to the terminal is (N). Therefore, the command

```
-COMPILE FILEB(N)↵
```

instructs the FORTRAN processor to eliminate the individual error messages for incorrect statements in the FILEB program when it compiles the program.

For a list of FORTRAN IV compiler switches, see the *Tymshare TYMCOM-X FORTRAN IV Reference Manual*.

Loader Switches

The switches described below are available for the commands LOAD, EXECUTE, CDEBUG, FDEBUG, DEBUG, and TRY. Occasionally, the user may want to set switches to direct loader operations. A loader switch consists of one letter, or a sequence of digits and one letter, preceded by a percent sign (%). A loader switch is inserted immediately before the file name to which it applies except for the %U switch, described below.

Common loader switches are:

Switch	Function
%S	Loads with symbols
%*n*O	Sets the program origin to *n* octal
%F	Causes an early search of the FORTRAN library
%P	Prevents a FORTRAN library search
%U	Prints all globals that are undefined after the files preceding the switch in the file list have been loaded

For example, the command

```
-LOAD F1, F2 %U, F3↵
```

instructs the loader to print a list of all undefined globals as soon as F1 and F2 are loaded.

Extended Command Forms

The user can employ more complex command forms for the compile-type commands as described below. These extended command forms allow the user to concatenate files during compilation, change a binary file name from its default name, and compile each of several files with the same file.

Concatenating Files in Compilation

The user may request that the system produce one relocatable file from a group of source files with any of the compile-type commands. In other words, the source files are compiled as if one large source file contained the concatenated text of all the individual files. He specifies this with the + construction. For example, the following command compiles the files PAR.F4, MAIN1.F4, and MAIN2.F4, producing one relocatable file, then loads and executes the file.

```
-EXECUTE PAR+MAIN1+MAIN2↵
```

The system assigns the name of the last file in the specification as the file name for the output file and appends the REL extension. In the example given above, the name of the output file is MAIN2.REL.

Changing Default Binary File Names

Usually, the relocatable file has the same name as the source file. The = construction allows the user to specify a file name other than the source file name for the relocatable file. The command form is

```
-COMPILE binary file name = source file name↵
```

For example, the command

```
-COMPILE BIN6 = PGM6↵
```

compiles PGM6 and writes the relocatable binary file with the name BIN6.REL.

The user may combine the + construction with the = construction. The following command assigns the name WHOLE.REL to the relocatable file produced by compiling PAR.F4, MAIN1.F4, and MAIN2.F4, and loads the relocatable file into core.

```
-LOAD WHOLE = PAR+MAIN1+MAIN2↵
```

Translations Using Parameter Files

The user may specify a group of programs, each of which he wants to compile with the same file, by using the <> construction.

NOTE: The + must always precede the <> construction.

For example, to compile AFILE, BFILE, and CFILE, each with the file PARAM, the user types

```
-COMPILE PARAM+<AFILE,BFILE,CFILE>↵
```

This is equivalent to the command

```
-COMPILE PARAM+AFILE,PARAM+BFILE,PARAM+CFILE↵
```

Both commands create the following relocatable files: AFILE.REL, BFILE.REL, and CFILE.REL.

ADDITIONAL DEBUGGING AIDS

The commands described below help the user examine his program closely and provide extensive information about the symbols and subprograms so the user can debug his program more easily.

The DDT Command

The DDT command allows the user to access DDT when his program(s) and DDT are already in core as the result of a previous TRY, DEBUG, GET, or RUN command. The form of the command is

-DDT↵

In the example shown below, the user executes the program PGM with the TRY command and encounters a program error; then he reenters DDT and examines the error.

```
-TRY PGM↵
MACRO: MULT
LOADING
EXECUTION

ILL MEM REF AT USER 005615

-DDT↵


5615←BEG
BEG/    MOVE AC,202400
.
.
.
```

The D Command

The D command stores (or deposits) information in the user's high or low segment area.[1] The form of the command is

-D left right [address]↵

left — Indicates the octal value to be stored in the left half of the location.

right — Indicates the octal value to be stored in the right half of the location.

address — Specifies the address of the location into which information is to be stored. If this argument is omitted, the information is stored in the location following that specified in the last D or E command.

1 - For a description of high and low segments of core, refer to the *DECsystem10 Assembly Language Handbook.*

The E Command

The E command examines a specified core location in the user's high or low segment area, printing the contents in half-word octal form. The form of the command is

-E [address]↵

where the address specifies an octal core location. If the address is omitted, the command examines the next sequential core location. For example,

```
-E 1170↵
001170/ 254000 001175    -E↵
001171/ 603000 000040    -E↵
001172/ 254000 001253    -
```

The CROSS Command

The CROSS command produces a cross-reference listing for every file compiled using the /CREF or /CROSS switch since the last CROSS command was executed or since logging in; then it deletes all files with the CRF extension. The form of the command is

-CROSS↵

The cross-reference listing is printed on the line printer unless the user specifies another device with the ASSIGN command. If the user specifies that DSK replace the line printer, the listings are written on a disk file with the same name as the source file and the extension LST.

In the example shown below, the user compiles TRIG with the /CREF switch and causes a cross-reference listing to be written on the file TRIG.LST.

```
-COMPILE TRIG/CREF↵
F-IV:  TRIG.F4
 MAIN.

EXIT

-ASSIGN DSK LPT↵
DSK ASSIGNED

-CROSS↵
```

```
-TYPE TRIG.LST
TRIG.F4 F40     V26(14) 1-MAY-74 13:55   PAGE 1


     1                          WRITE (5,10)
     2                          READ (5,100) ANGLE
     3                          X=SIN(ANGLE)
     4                          Y=COS(ANGLE)
     5                          WRITE(5,20)X,Y
     6                 10       FORMAT (14H ENTER ANGLE: )
     7                 20       FORMAT (6H SIN= ,F5.2,5X,6H COS= ,F5.2)
     8                 100      FORMAT(F7.2)
     9                          END

        SUBPROGRAMS

        FORSE.  JOBFF   SIN     COS     FLOUT.  FLIRT.  EXIT

        SCALARS

        ANGLE   45              X       46              Y       47

         MAIN.



SYMBOL CROSS REFERENCE

ANGLE           2       3       4
COS             4
SIN             3
X               3       5
Y               4       5



MACRO/OPDEF CROSS REFERENCE

   10P          1       6
   20P          5       7
  100P      100P        2       8
-
```

INITIATING OR CONTINUING EXECUTION

The commands described below begin execution of the user's program(s) or continue execution after an interruption.

The START and CSTART Commands

The START command begins execution of the program(s) currently in core. The form of the START command is

-START↵

In the example shown below, the user compiles and loads the program PGM and begins execution.

```
-LOAD PGM↵
F-IV:  PGM
 MAIN.
LOADING
6K CORE

-START↵


EXECUTION COMPLETE
EXIT

-
```

The CSTART command begins execution as does the START command, but permits certain XEXEC commands to be entered while the program is running.[1] The commands that are allowed are commands such as TIME, DETACH, etc., which do not destroy the user's core image or begin execution.[2] For example,

```
-LOAD TRPRG, SC↵
F-IV:  SC
 SC
LOADING
6K CORE

-CSTART↵                    The user begins execution of the loaded program.

-TIME↵                      He executes the TIME command while his program is being executed.
11.83 TRU
35.97 TRU
TERMINAL TIME: 0:24:01

-
```

1 - See page 67 for information about the use of the CSTART command in detached processing.
2 - Commands that do not destroy the current core image are indicated in Appendix A, page 85.

If a command is entered that would destroy the user's core image, the system prints

PLEASE TYPE ESC FIRST

The user can interrupt execution initiated by CSTART by typing one Alt Mode/Escape or by typing the HALT command. A program being executed by the CSTART command cannot receive input from the terminal. Execution is suspended if the program requests input from the terminal.

The CONTINUE and CCONTINUE Commands

The CONTINUE command resumes execution of an interrupted program. The form of the command is

-CONTINUE↵

If the user had typed two Alt Mode/Escapes during program execution to interrupt execution and return to XEXEC, he could subsequently (1) enter the CONTINUE command to resume execution immediately or (2) enter any XEXEC command(s) that do not destroy the current core image before continuing execution.[1]

NOTE: Typing Alt Mode/Escapes clears the terminal output buffer; some of the program's output may be lost during this process.

In the example given below, the user loads and begins execution of TLOOP, interrupts execution, and then continues execution using the CONTINUE command.

```
-EXECUTE TLOOP↵                    The user begins execution of TLOOP, which prints data on the terminal.
LOADING
EXECUTION

  0.84          0.54
  0.91         -0.42
  0.14         -0.99
⊕ ⊕                                The user types two Alt Mode/Escapes to interrupt the execution.
-TIME↵                             The user checks the number of TRUs the program has used.
 3.01 TRU
40.90 TRU
TERMINAL TIME: 0:24:01

-CONTINUE↵                         He types the CONTINUE command to continue execution of
                                   TLOOP from the point of interruption.
 -0.43         -0.90
 -0.99         -0.13
 -0.64          0.77
  0.30          0.96
  0.96          0.27
  0.75         -0.67
   .
   .
   .
```

1 - The XEXEC commands that do not destroy the current core image are indicated in Appendix A, page 85.

The CCONTINUE command is identical to the CONTINUE command except that CCONTINUE allows the user to enter commands that do not destroy the current core image while the program is running.[1]

```
-EXECUTE NPLOOP↵
LOADING
EXECUTION
⊕⊕                              The user interrupts execution of NPLOOP.
-CCONTINUE↵                     He continues execution of NPLOOP.

-WHO↵                           He tries to execute a command that would destroy the core image.
PLEASE TYPE ESC FIRST

-DAY↵                           The DAY command does not destroy the core image so it can be
1-MAY-1974 15:03:01             executed while NPLOOP is running.

-

ENTER THE NEW COUNT:            NPLOOP requests input from the user.

33↵                             The user types the requested data.

?33?                            The system is at XEXEC command level and attempts to interpret
                                33 as a XEXEC command.
-⊕
                                The user types one Alt Mode/Escape and the CONTINUE command
-CONTINUE↵                      so he can enter data into his program from the terminal.

33↵

THE AVERAGE FOR A POPULATION OF 33 = 2.77

-
```

CORE IMAGE FILES

When a user loads a program into core, the entire environment needed for the execution of the program is loaded into core as well. The program and its environment, or the core image of the loaded program, can be saved on a disk file so that it is not necessary to load the program and associated subprograms each time it is run. The commands presented below create or use core-image files.

The SAVE and SSAVE Commands

The SAVE command stores on a file the image of the program and environment currently loaded in core. The form of the SAVE command is

-SAVE file identifier↵

1 - The XEXEC commands that do not destroy the current core image are indicated in Appendix A, page 85.

If the user does not specify an extension in the file identifier, the system assigns the SAV extension. If the user has divided his program into two segments, the system assigns the HGH extension to the high segment and the LOW extension, or the extension given by the user, to the low segment. In the example shown below, the user loads the program PGM5 and saves the core image of the program with the name PGM5.SAV.

```
-LOAD PGM5↵
LOADING
6K CORE

-SAVE PGM5↵
JOB SAVED

-
```

The SSAVE command is identical to the SAVE command except that SSAVE assigns the SHR extension to the high segment of a program if a high segment exists. The high segment of the program is then sharable with other users. As with the SAVE command, SSAVE assigns the LOW extension or the user-specified extension to the low segment of a program divided into two segments.

The GET, RUN, and GO Commands

The GET command loads a core-image file but does not begin execution.[1] The form of the command is

-GET file identifier↵

If the user does not specify an extension in the file identifier, the extension is assumed to be SAV, HGH and LOW, or SHR and LOW.

The RUN and GO commands load a core image file and begin execution. The forms of the commands are

-RUN file identifier [*n*]↵

where *n* is the maximum core to be allowed for program execution, and

-GO file identifier↵

If the user does not specify an extension in the file identifier, the extension for the low segment is assumed to be SAV or LOW. If there is a high segment, the extension is HGH or SHR. In the example shown below, the user loads the file MAIN and all subroutines called by MAIN, and saves the core image of MAIN and the associated subprograms on the file PACKG.SAV.

```
-LOAD MAIN,FRONT,BD,COMNEW↵
LOADING
8K CORE

-SAVE PACKG↵
JOB SAVED

-
```

1 - Execution may be initiated by the START command, or DDT may be accessed with the DDT or REENTER command after the core-image file is loaded.

The core image of the loaded program and associated subprograms is now stored on the file PACKG; henceforth, it can be executed simply by typing

-RUN PACKG↵

RELEASING FILES

The CLOSE command allows the user to terminate input and output and close all files when a program has been interrupted with Alt Mode/Escapes. The form of the command is

-CLOSE [device name]↵

The CLOSE command terminates input and output on all devices or on the specified device. The CLOSE command preserves all device assignments.

The FINISH command is identical to the CLOSE command except that FINISH releases device assignments. The form of the command is

-FINISH [device name]↵

The FINISH command terminates input and output on all devices, and all device assignments are released; or, if a device is specified, the command terminates input and output and releases assignments of the specified device.

DETACHED PROCESSING

Detached processing enables the user to free his terminal from a job while it is running. Thus, the user can begin execution of a job and detach the job from the terminal leaving him free to log in again and begin execution of another job or attach to a previously detached job. The commands used for detached processing are described in the table below.

Command	Function
CCONTINUE	Continues execution of an interrupted program and at the same time allows the user to type certain commands at the terminal. See page 64 for more information
CSTART	Begins execution of the user's program in core and allows the user to type certain commands at the terminal. See page 63 for more information
DETACH	Detaches the terminal from the current job. It is usually preceded by CSTART or CCONTINUE. If a detached job requests terminal input or output, execution of the program is suspended until a terminal is attached to the job again
ATTACH	Detaches the user from his current job and attaches him to another running job

Two methods for detaching a job from the terminal are described below.

Method A	Method B
1. The user loads his program into core.	1. The user begins execution of his program.
2. He types the CSTART command to begin execution of the program while allowing him to type other commands.	2. He interrupts the program by typing two Alt Mode/Escapes. 3. He uses the CCONTINUE command to continue execution of the program while allowing him to type other commands.
3. He types the DETACH command to detach the running job from the terminal.	4. He types the DETACH command to detach the running job from the terminal.

When the user detaches his job from the terminal, he is logged off automatically. When he logs in again, he can attach the terminal to the detached job or he can log in to a new job.

The example given below illustrates the use of detached processing.

```
PLEASE LOG IN:  SUSAN;;↵          The user logs in under user name SUSAN.

TYMSHARE  1050  10-MAY-74

-EXECUTE LPROG↵                   The user begins execution of her job.
LOADING
EXECUTION
⊕ ⊕                               She types two Alt Mode/Escapes to return control to XEXEC.
-CCONTINUE↵                       The CCONTINUE command continues execution of the interrupted program
                                  and allows the user to type certain commands at the terminal as well.
-DETACH↵                          The DETACH command prints the number of the running job and returns the
FROM JOB 4                        user to TYMNET. The program continues execution.
```

```
PLEASE LOG IN:  SUSAN;;↵
TYPE JOB NUMBER TO ATTACH, JOB 4 OR C.R. TO LOGIN↵

TYMSHARE 1051 10-MAY-74

-GET PGM5↵
JOB SETUP

-CSTART↵

-DETACH↵
FROM JOB 6


PLEASE LOG IN: SUSAN;;;4↵

-
EXIT

-ATTACH 6↵
FROM JOB 4

-
PGM5 PROCESSING COMPLETE
EXIT

-LOGOUT↵

ANOTHER JOB STILL LOGGED IN UNDER SUSAN
2.87 TRU
TERMINAL TIME: 0:01:01

PLEASE LOG IN:
```

The user types a Carriage Return to log in under a new job.

The user loads a core image file into core.

The CSTART command begins execution of the user's program in core and allows the user to type certain commands at the terminal as well.

The user detaches the terminal from job 6.

The user logs into job 4, which is running in the detached mode.

The program running under job 4 runs to completion.

The user attaches job 6 to the terminal.

The user logs out.

The system notifies the user that another job (job 4) is still logged in under user name SUSAN, prints the system resources used for the completed job, and returns control to TYMNET.

Section 7
SYSTEM INFORMATION AND CONTROL

The commands described in this section allow the user to print information about the system or about his job(s), to change the assignments of devices, and to change operation modes.[1]

PRINTING SYSTEM AND JOB INFORMATION

The commands described below provide information about the current status of the system and the user's job(s). The information printed by some of these commands is determined by the license and status of the user. For example, an Account Supervisor has access to information about all users in his account, whereas a typical user has access to information about his own jobs only. The different levels of information available from each command are described in the following subsections.

The CORE Command

The CORE command prints the amount of core assigned to the user's job and the amount of unused core available. This information is printed in the form

m+*n*/*p* CORE
VIR. CORE LEFT=*v*

m Indicates the number of blocks in the low segment; *m* is in units of 1000 blocks.[2]

n Denotes the number of blocks in the high segment; *n* is in units of 1000 blocks.[2]

p Shows the amount of core available to this job.

v Indicates the number of blocks of unused core available to the user's job; *v* is in units of 1000 blocks.[2]

For example,

```
-CORE↵
6+0/50K CORE
VIR. CORE LEFT=736

-
```

The command

-CORE *n*↵

changes the amount of core assigned to the user's job to *n*. If *n* is 0, the command clears the low and high segments of core assigned to the user's job.[3]

1 - The TYMCOM–X operation modes are presented on page 79.
2 - A block consists of 1024 computer words.
3 - For a description of low and high segments of core, refer to the *DECsystem10 Assembly Language Handbook.*

The DATE and DAYTIME Commands

These commands print the calendar and clock information. The DATE command prints the time since log in or since the last DATE command. The DAYTIME command always prints the time since log in. For example,

```
-DATE↵
MAY 7, 1974  15:57
-DAYTIME↵
7-MAY-74 15:57:12

-
```

The DSK Command

The DSK command prints (1) the number of 128-word disk blocks read and written since the last DSK command, or since log in if there has been no DSK command in the current session, and (2) the total number of disk blocks read and written since log in. For example,

```
-DSK↵
RD,WT=25,2
RD,WT=31,6

-
```

The PJOB Command

The PJOB command prints the number of the job to which the user's terminal is currently attached. For example,

```
-PJOB↵

13

-
```

The PPN Command

The PPN command prints the global account number (GAN) and the file directory number for the specified user name. For example,

```
-PPN↵
USER NAME: WMS↵
12051,105547
-
```

or

```
-PPN WMS↵
12051,105547
-
```

The RESOURCES Command

The RESOURCES command prints the names of all input and output devices available to the user. For example,

```
-RESOURCES↵
DSKB,LPT
-
```

The disk and the line printer are available to the user.

The SET LIMIT Command

The SET LIMIT command allows the user to limit the TRUs used. For example,

```
-SET LIMIT 20↵
-RUN PROG↵
TIME LIMIT EXCEEDED
-
```

The user sets a limit of 20 TRUs.

He executes a program.

The program is interrupted when 20 TRUs are used. The user can reset the TRU limit and continue, or he can abort the run.

The SYSNO Command

The SYSNO command prints the computer system number and the current version number of the monitor. For example,

```
-SYSNO↵
TYMSHARE C33-P012/M 4-18-74
-
```

Computer system 33 — *Version of monitor* — *Release date of this version of the monitor*

The SYSTAT Command

The SYSTAT command informs the user of the status of his jobs. For example,

```
-SYSTAT↵

TYMSHARE C33-P012/M 4-18-74 AT 15:57:59 ON WED MAY 1,1974
10 JOBS IN USE OUT OF 59.

JOB  TYMNET PROG   SIZE   STATE       TRU  CONNECT   USER NAME
19* AME1 63D SYSTAT 3+6   RN         0.90  0:00:12   AUSTIN $
$ MEANS RUN ONLY PROGRAM

PROGRAM HIGH(K) USERS   OWNER
SYSTAT   6      1       SYS


-
```

If the user is an Account Supervisor, the SYSTAT command prints the above information for all users in the account. For example,

```
-SYSTAT↵

TYMSHARE C33-P012/M 4-18-74 AT 17:30:41 ON WED MAY 1,1974
4 JOBS IN USE OUT OF 59.  1 DETACHED

JOB  TYMNET PROG   SIZE   STATE       TRU  CONNECT   USER NAME
 9* AME1 73D SYSTAT 3+6   RN        8.08  0:02:53   COLLINS $
15 DETACHED LOGINN 1+0    ^C        0.98  0:00:36   SFS
17 AME1 65D WHO    3+6    ^C        1.75  0:00:53   SFS $
18 WC02 64D STAR   5      ^C        1.40  0:00:37   PARK
$MEANS RUN ONLY PROGRAM

PROGRAM HIGH(K) USERS   OWNER
SYSTAT   6      2       SYS


-
```

The SYSTAT command provides many options for printing the requested information. Some of the more popular options are shown below. The user can include as many options as he chooses in a SYSTAT command.

Option	Meaning	Sample SYSTAT Command
CONTINUOUS	Displays the requested information repeatedly so that the information can be observed as it is being changed by the program	-SYSTAT CONTINUOUS↵
<program>	Prints information about all jobs running the specified program	-SYSTAT <STATPAK>↵
(user name)	Prints information about all jobs running under the specified user name	-SYSTAT (SFS)↵
	Prints information about the user's current job	-SYSTAT .↵
ME	Prints the job number and user name of the current user	-SYSTAT ME↵
EVERYTHING	Prints all system information available to the user	-SYSTAT EVERYTHING↵
?	Prints operating instructions for SYSTAT and all available options	-SYSTAT ?↵

The TIME Command

The TIME command prints the number of TRUs used since the last TIME command and since log in; it also prints the total connect time used. For example,

```
-TIME↵
1.36 TRU                    TRUs since the last TIME command.
40.07 TRU                   Total number of TRUs consumed since log in.
TERMINAL TIME: 0:38:55      Total connect time.

-
```

The USERS Command

The USERS command prints a summary of information about jobs on the system. For a typical user, the USERS command prints information about the jobs logged in under his user name; for an Account Supervisor, the command prints information about all the jobs logged in under his account. The summary is in the following form:

USR=*i* RUN=*j* DIO=*k* TIO=*l* IO=*m* SPC=*n* DET=*p* [*q*,*r*]

i Indicates the number of jobs logged in.

j Specifies the number of jobs running.

k Indicates the number of jobs waiting for disk input or output.

l Denotes the number of jobs waiting for terminal input or output.

m Indicates the number of jobs waiting for input or output on devices other than a disk file or the terminal.

n Shows the number of special queues being used.

p Indicates the number of detached jobs.

q Denotes the number of people logged in within the user's global account.

r Indicates the number of jobs logged in under the current user name.

For example,

```
-USERS↵
USR=11 RUN=2 DIO=0 TIO=8 IO=2 SPC=2 DET=0  [1,1]
```

The VERSION Command

The VERSION command prints the version number of the program currently in core. For example, to determine the version of STATPAK on the system, the procedure is

```
-GET SYS:STATPAK↵        The user loads the program with the GET command.
JOB SETUP

-VERSION↵                The current version number of STATPAK is printed on the terminal.
421441 444415
-
```

The WATCH Command

The WATCH command causes time or disk information to print automatically when execution of a program or command is begun or halted. This command provides the user with a tool for measuring the performance of his program. The form of the WATCH command is

-WATCH [arguments]↵

arguments The arguments included in the command determine what information is printed when program execution is begun or halted. Multiple arguments are separated by commas or spaces. The argument DAY prints the time of day when control passes to a user program or a XEXEC command. The following arguments print information for the command or the program executed whenever control returns to XEXEC:

- READ Prints the number of disk blocks read, modulo 4096.
- RUN Prints the elapsed run time which is the time expended in actual execution.
- WAIT Prints the wait time which is the time expended while waiting for execution to commence.
- WRITE Prints the number of disk blocks written, modulo 4096.

If no arguments are included, the command cancels any existing WATCH options.

NOTE: Each WATCH command overrides the previous WATCH command.

In the example shown below, the user requests that the system print the time of day and the run time for each XEXEC command; he then enters the USERS and SYSNO commands. The user terminates the previous WATCH command by typing WATCH followed by a Carriage Return. Finally, he enters the SYSNO command.

```
-WATCH DAY, RUN↵

-USERS↵
[16:18:54]        The time of day and run time are printed with each command.
USR=26 RUN=2 DIO=1 TIO=21 IO=0 SPC=2 DET=0  [1,1][2.21    ]

-SYSNO↵
[16:19:06]        Time of day.
TYMSHARE C33-P012/M 4-18-74[0.37    ] Run time.

-WATCH↵           The user cancels the previous WATCH command.

-SYSNO↵
TYMSHARE C33-P012/M 4-18-74  The time of day and run time are not
-                            printed for the SYSNO command.
```

The WHERE Command

The WHERE command prints the job number to which the specified user is attached. The form of the command is

-WHERE user name↵

For example,

```
-WHERE CHAMBERLAIN↵

ON JOB  8
-
```

The WHO Command

For a typical user, the WHO command prints the job number of each currently active job under his user name. For example,

```
-WHO↵
19* RUTZ

-
```

If the user is an Account Supervisor, the WHO command prints the job numbers and user names for all jobs logged in under his account. In the example below, an Account Supervisor enters the WHO command.

```
-WHO↵
 9* NEWLY        An asterisk follows the job number of the user executing the WHO command.
15  SFS
17  SFS
18  PARK

-
```

DEVICE ASSIGNMENTS

The user can change the assignments of the terminal, line printer, or disk from their default assignments; the default assignments vary with different commands. For example, the default device assignment for the TYPE command is the terminal and the default device assignment for the LIST command is the line printer. The device assignments can be changed using the ASSIGN and DEASSIGN commands, described below.

The ASSIGN Command

The ASSIGN command changes a device assignment. The form of the command is

-ASSIGN $device_1$ $device_2$↵

$device_1$ Indicates the desired device name.

$device_2$ Indicates the name of the device to be replaced.

The device names that can be entered are TTY for terminal, DSK for disk, and LPT for line printer.

For example, the command

-ASSIGN TTY LPT↵

causes all commands and programs that normally print information on the line printer to write the information on the terminal instead.

The DEASSIGN Command

The DEASSIGN command ends device assignments made by the ASSIGN command. The form of the command is

-DEASSIGN [device]↵

where device specifies the name of the device to be deassigned. If no device is specified, the command ends all device assignments.

For example,

-DEASSIGN TTY↵

ends the assignment of the terminal made by a previous ASSIGN command.

OPERATION MODES

There are two operation modes available on the TYMCOM-X besides XEXEC. They are the PDP-10 and SUDS modes. If the user should want to access one of these modes from XEXEC or transfer between modes, he simply types the appropriate command to enter the operation mode, as detailed below.

To enter XEXEC from PDP-10 or SUDS, the user types XEXEC and a Carriage Return. To enter PDP-10 from XEXEC or SUDS, the user types PDP10 and a Carriage Return. To enter SUDS from XEXEC or PDP-10, the user types HELLO and a Carriage Return.

Section 8
INDIRECT COMMANDS

XEXEC provides two methods for executing commands indirectly: the user can store file names and switches on a file, called a command string file, and then use the command string file name in certain commands, or the user can store commands on a file and then execute them one by one using the PERFORM command.

COMMAND STRING FILES

When the user wants to specify the same group of files and switches for a number of commands, he may store the information on a command string file to avoid typing it with each command. Then, when entering the command, the user simply types the name of the command string file preceded by the character @. Command string files can be used with any XEXEC command that allows multiple file names and switches. These commands are COPY, DECLARE, DELETE, DIRECTORY, FILES, LIST, PRINT, TYPE, RENAME, CDEBUG, COMPILE, DEBUG, EXECUTE, FDEBUG, LOAD, and TRY.

For example, if the file ROUT3 contains the string

PROG2/LIBRARY,PROG3,PROG4

the command

-COMPILE PROG1,@ROUT3,PROG7↵

is equivalent to

-COMPILE PROG1,PROG2/LIBRARY,PROG3,PROG4,PROG7↵

The command string file identifier may have any extension. If the extension is CMD or there is no extension, the user can refer to the file in a command by typing the file name only; otherwise, the user must include the extension with the file name.

The user can write a multiline command string file by using a semicolon (;) to continue onto the next line. For example, the command string file above can also be written as follows:

```
PROG2/LIBRARY,;
PROG3,PROG4
```

As all text on a line after a semicolon is ignored, the user can include comments in a command string file after the semicolon. For example, the file INV.CS contains the lines

```
INV.SEP,PGM1,NEWDAT.DAT;NEWDAT IS NEW DATA
;INVENTORY PROGRAMS
```

The command

-DELETE @INV.CS↵

deletes the files INV.SEP, PGM1, and NEWDAT.DAT.

The user may nest command string files within command string files; as many as nine levels of nesting are allowed. For example, if the file AC1.CMD contains the string

ACCT.A1,@INV.CS

the command

-LIST @AC1↵

lists the contents of ACCT.A1, INV.SEP, PGM1, and NEWDAT.DAT on the line printer.

THE PERFORM COMMAND

The PERFORM command instructs the system to read XEXEC commands from a specified file and execute them one by one. The form of the PERFORM command is

-PERFORM file identifier↵

where the file identifier specifies the file on which the XEXEC commands are stored. This file is called a PERFORM file. When the user enters the PERFORM command, the system reads the commands from the PERFORM file and begins executing them in order. If a command in the PERFORM file requires terminal input such as a Y or N to confirm file deletion, the system pauses for the appropriate input from the terminal. Terminal input or responses must not be included in the PERFORM file itself, but have to be typed at the terminal.

Each command in a PERFORM file must be entered on a separate line and should not be abbreviated. Any text, switches, or notations which would normally be allowed in the commands can be included in the PERFORM file. The XEXEC commands that may be used in a PERFORM file are listed below.

Legal Commands in a PERFORM File

The XEXEC commands that may appear in a PERFORM file may be divided into four groups: commands that return control to PERFORM, commands that transfer control to a subsystem, commands that execute a program, and commands that terminate the user's TYMCOM-X session.

Commands That Return Control to PERFORM

A command that returns control to PERFORM may appear anywhere in the PERFORM file. The commands in this group are:

COMPILE	**DATE**	**DIFFER**	**FILES**	**PDP10**	**PRINT**	**XEXEC**
COPY	**DECLARE**	**DIRECTORY**	**HELP**	**PFDC**	**RENAME**	
CROSS	**DELETE**	**FDC**	**LIST**	**PPN**	**TYPE**	

If the last command in the PERFORM file is one of the commands listed above, the system executes the command, prints

END PERFORM JOB

and returns control to XEXEC.

For example, the user employs the PERFORM command and a PERFORM file to print a list of all files created since the seventh day of the current month, with their sizes and creation dates, and an alphabetical directory listing of the files that have the extension F4.

```
-TYPE P11↵                                    The user prints his PERFORM file.
FILES /AFTER 7 /SIZE /CREATION
DIRECTORY *.F4 /ALPHABETICAL

-PERFORM P11↵

P11                    5  10-MAY-74
TRIG      REL          5  10-MAY-74
PGM5      SAV         40   7-MAY-74      The FILES command produces this list.
SUBODD    F4          15   7-MAY-74
BIGFIL                 5   7-MAY-74

SINCOS    F4           5  26-APR-74 1215      ALL RD  NO     28-APR-74(A)
SUBODD    F4          15   7-MAY-74 1455      ALL RD  NO      7-MAY-74(A)
TRIG      F4           5   7-MAY-74 1627      ALL RD  NO     10-MAY-74(A)
                                                The DIRECTORY command produces this list.
END PERFORM JOB

-
```

Commands That Transfer Control to EDITOR or a Conversational Language

A command that transfers control to EDITOR or a conversational language terminates execution of the PERFORM file; therefore, it must appear as the last command in the PERFORM file. The commands in this group are:

CREATE	**MODIFY**	**CFORTRAN**
EDITOR	**XBASIC**	⋮
		(any conversational language)

Commands That Execute a Program

A command that executes a program usually appears as the last command in the PERFORM file since an executed program does not return control to the PERFORM file unless it is specifically designed to do so. The commands in this group are:

EXECUTE	**R**	**CDEBUG**
TRY	**RUN**	**FDEBUG**

The user may design a FORTRAN, COBOL, or MACRO program to return control to the PERFORM file rather than to XEXEC; a command that executes such a program may appear anywhere in the PERFORM file.

In FORTRAN, control is returned to the PERFORM file by the statement

CALL EXITPE

For example, the following FORTRAN program returns control to a PERFORM file at the end of execution

```
      ACCEPT 20,A
20    FORMAT(E12.5)
      B = A**.25
      TYPE 20,B
      CALL EXITPE
      END
```

In COBOL, control is returned to a PERFORM file by the subroutine call

ENTER MACRO EXITPE

In MACRO, control is returned to a PERFORM file by a RUN UUO consisting of the following instructions:[1]

```
        MOVE    AC,[1,,RUNBLK]
        RUN     AC,
        HALT    .-1
        :
RUNBLK: SIXBIT/SYS/
        SIXBIT/PERFOR/
        EXP 0,0,0,0
```

The user may include statements that return control to XEXEC and statements that return control to a PERFORM file in the same program. If no PERFORM file exists, either type of statement returns control to XEXEC.

Commands That Terminate the User's TYMCOM-X Session

Any of the following commands can be included in a PERFORM file to terminate the user's TYMCOM-X session:

EXIT **KJOB** **LOGOUT**

These commands are described on pages 6 and 7.

Interrupting the PERFORM Command

The user may interrupt execution of the commands in a PERFORM file by typing two Alt Mode/Escapes to return control to XEXEC. The interruption leaves the PERFORM file active; thus, the user can return to the point at which the interruption occurred by typing the REENTER or CONTINUE command provided that the command he interrupted allows him to do so.

NOTE: If the user interrupts a PERFORM command and then executes a program that is designed to return control to a PERFORM file, control returns from the executed program to the interrupted PERFORM file even though the program itself was not executed by this PERFORM file.

1 - See the *DECsystem10 Assembly Language Handbook* for a discussion of machine language programming and UUOs.

Appendix A
XEXEC COMMAND SUMMARY

The table below lists the XEXEC commands in alphabetical order. The format and description of each command is given, as well as information about the command's effect on core and the number of the page on which the command is fully documented.

Command	Destroys Current Core Image	Description	Page
ASSIGN $device_1$ $device_2$		Changes a device assignment from $device_2$ to $device_1$	78
ATTACH *n*	•	Attaches the terminal to job *n*	67
CCONTINUE	•	Continues program execution from point of interruption and permits concurrent execution of XEXEC commands that do not destroy the core image	64
CDEBUG [file list]	•	Compiles the specified file(s) if necessary and loads the file(s) and the COBOL debugger	54
CLOSE [device name]	•	Terminates input and output on the specified device or on all devices	67
COMPILE [file list]	•	Compiles specified file(s)	52
CONTINUE	•	Continues execution from point of interruption	64
COPY source $\left\{\begin{matrix}\text{TO} \\ , \end{matrix}\right\}$ destination	•	Writes a copy of the source file on the destination file	22
CORE [*n*]		Changes the amount of core assigned to the user's job to *n*. If *n* is not specified, prints the amount of core assigned to the user's job and the amount of unused core available	71
CREATE file identifier	•	Transfers control to EDITOR to create a new file	45
CROSS	•	Produces cross-reference listing for all files compiled with the /CREF or /CROSS switch	61

(Table continues)

Command	Destroys Current Core Image	Description	Page
CSTART	•	Begins program execution and permits concurrent execution of XEXEC commands that do not destroy the core image	63
D left right [address]		Stores information in the left half and right half of the specified location	60
DATE	•	Prints date and time to the minute	72
DAYTIME		Prints date and time to the second	72
DDT	•	Calls DDT debugger	60
DEASSIGN [device]		Ends assignment of specified device or of all devices	78
DEBUG [file list]	•	Compiles the specified files(s) if necessary and loads the file(s) and the DDT debugger	55
DECLARE private account public file name(s)	•	Sets protection for specified file(s)	37
DELETE file list	•	Deletes specified file(s)	24
DETACH		Detaches the terminal from the current job	67
DIFFERENCES	•	Compares two files and lists all differences	31
DIRECTORY [file list]	•	Prints descriptive information about the specified file(s) or all files in the user's directory	41
DSK		Prints number of disk blocks read and written since last DSK command and since log in	72
E [address]		Prints contents of specified core location	61
EDITOR	•	Transfers control to EDITOR	45
EXECUTE	•	Compiles, if necessary, loads, and executes the specified files	53
EXIT	•	Terminates the TYMCOM-X session without printing TRUs and connect time used	7
FDC	•	Sets general protection for access to user's file directory	35

Command	Destroys Current Core Image	Description	Page
FDEBUG [file list]	•	Compiles the specified file(s), if necessary, and loads the file(s) and the FORTRAN debugger	54
FILES [file list]	•	Prints specified file names or all file names	41
FINISH [device name]	•	Terminates input and output on the specified device and releases the device assignment, or performs this function for all devices	67
GET file identifier	•	Loads the specified core-image file into core	66
GFD user name	•	Accesses the directory of another user in the same account if license permits	44
GO file identifier	•	Loads and executes the specified core-image file	66
HELLO	•	Transfers control to SUDS mode	79
HELP	•	Prints a list of XEXEC commands	10
KJOB	•	Terminates TYMCOM-X session and prints TRUs and connect time used	6
LIST file list	•	Prints the specified files on the line printer with page headings	27
LOAD [file list]	•	Compiles, if necessary, and loads the specified files	53
LOGOUT	•	Terminates TYMCOM-X session and prints TRUs and connect time used	6
MODIFY [file identifier]	•	Transfers control to EDITOR and reads the specified file, or the file specified in the most recent CREATE or MODIFY command	45
PDP10	•	Transfers control to PDP-10 mode	79
PERFORM file identifier	•	Executes commands stored on the specified file	82
PFDC	•	Prints general protection for access to user's file directory	36

(Table continues)

Command	Destroys Current Core Image	Description	Page
PJOB		Prints the current job number	72
PPN user name	•	Prints the global account number (GAN) and the file directory number for the specified user name	73
PRINT file list	•	Prints the specified files on the terminal without headings	27
R program name	•	Executes specified system program	47
REENTER	•	Returns control to interrupted program, preserving the user's work	46
RENAME old file name {AS / ,} new file name	•	Changes the name(s) of the specified file(s)	26
RESOURCES	•	Prints names of available input and output devices	73
RUN file identifier	•	Loads and executes specified core-image file	66
SAVE file identifier	•	Writes current core image on the specified file	65
SET LIMIT *n*		Interrupts execution when the number of TRUs consumed is *n*	73
SSAVE file identifier	•	Writes current core image on specified file making the high segment sharable	65
START	•	Begins execution of program currently in core	63
SYSNO	•	Prints system number, the current version number, and release data of the monitor	74
SYSTAT	•	Prints information about the status of the user's job or the status of all jobs in the account	74

Command	Destroys Current Core Image	Description	Page
TIME		Prints the number of TRUs consumed since the last TIME command and since log in, and prints the total connect time used	75
TRY [file list]	•	Compiles the specified file(s), if necessary, loads the files and the DDT debugger, and begins execution	55
TTY terminal feature		Sets specified terminal feature	11
TYPE file list	•	Prints the specified file(s) on the terminal	27
USERS	•	Prints information about all accessible files in the user's account	75
VERSION	•	Prints version number of the program currently in core	76
WATCH [argument(s)]		Prints requested information with each subsequent XEXEC command; arguments are DAY, READ, RUN, WAIT, and WRITE	76
WHERE user name		Prints job number(s) that the specified user is logged into	77
WHO	•	Prints job number and user name for all accessible jobs in the user's account	77
XEXEC	•	Transfers control to XEXEC mode	79

Appendix B
TERMINAL FILLER CLASSES

The filler classes for output and echoing are listed in the table below. These classes can be assigned to a terminal by using the TTY FILL command, described on page 11.

Character Name	Octal Code	Number of Fillers for Filler Class			
		0	1	2	3
BS	010	0	2	6	6
HT	011	0	1 *or* 2	1 *or* 2	1 *or* 2*
LF	012	0	1	6	6
VT	013	0	2	6	6
FF	014	0	12	21	21
CR	015	0	1 *or* 2	2 *or* 4	2 *or* 4**
XON	021	0	1	1	1
TAPE	022	0	1	1	1
XOFF	023	0	1	1	1
NTAP	024	0	1	1	1

*1 if 0–3 spaces to tab stop; 2 if 4–7 spaces to tab stop.
**1 or 2 if CR is typed; 2 or 4 if CR is supplied because the line is too long.

INDEX

NOTE: Page numbers which appear in **bold** *face type refer to those pages where the listed item receives the most detailed discussion.*