Georgia
Institute
of
Technology

RICH ELECTRONIC COMPUTER CENTER / (404) 894-3100 / ATLANTA, GEORGIA 30332

DEMAND TERMINAL USER'S MANUAL

FOR THE

UNIVAC 1108

May 1972

(Revised)

DEMAND TERMINAL USER'S MANUAL

FOR THE

UNIVAC 1108

May 1972

(Revised)

PREFACE

This manual is one of a series of manuals prepared by the Rich Electronic Computer Center for the benefit of its users. It is concerned with usage of demand terminals (Teletypes and other keyboard devices) when connected to Georgia Tech's UNIVAC 1108 computer. The executive system in use on the 1108 is called EXEC 8.

This manual is intended to satisfy two objectives. First, the manual provides an introduction to terminal operation and system concepts. No previous knowledge is assumed. Second, complete details on exclusively demand features of the system are given. Thus, each chapter is either an introductory, tutorial chapter or a reference chapter.

Chapters 1 and 2 provide information on how to connect to the 1108 and begin a run. All users need this information. Chapter 3 is for FORTRAN, COBOL, and ALGOL programmers and demonstrates use of these languages. A basic knowledge of one of the languages would be helpful. Chapter 4 is the reference chapter for the ED processor, a tool widely used from demand terminals. Chapter 5 describes system failures and how to be prepared for them. Chapter 6 is a reference chapter describing demand terminal operation. Features such as paper tape, the BREAK and TAB keys, and messages to and from the central site operator are discussed here. Chapter 7 is a tutorial showing how certain EXEC 8 features may be used to advantage. Finally, Chapter 8 describes several programs of general application written especially for demand use.

The primary reference for the system is the RECC publication, Programmers Reference Manual for the UNIVAC 1108, EXEC 8 Executive System (hereafter referred to as the GTPRM). It is hoped that sufficient information is presented here to avoid the necessity of referencing the GTPRM; however, users with special interest or unusual applications may need to do so. For this reason, there are numerous references to sections in the GTPRM that may be referenced for further detail.

The demand programmer will probably require one or more UNIVAC reference manuals describing a programming language, because the languages themselves are not discussed here. But, if a user is already familiar with FORTRAN, ALGOL, or ANSI COBOL, the examples presented here may provide sufficient information to permit him to run programs.

Much of the reference material contained in this manual was reprinted from UNIVAC documentation, with appropriate editing for the Georgia Tech environment.

TABLE OF CONTENTS

FIGURES

1. DEMAND TERMINAL BASICS

1.1. Definition of Demand Processing

Demand processing is defined as a mode of operation in which run processing is dependent on manual interface with the executive during processing. Basically, it is a conversational mode of operation requiring a demand and response type of activity. Conversational operation via a remote terminal causes the executive, a demand processor, or an active program to immediately react and respond. Demand processing terminals are generally thought of as being remote from the computer site and to have a printer or a cathode ray tube and a keyboard. An example of a demand terminal is the teletypewriter keyboard and printer.

The distinction between batch-mode processing and demand processing lies in the frequent interaction with the user that occurs during demand processing. The terminal user is considered to be in conversation with the executive, special demand function, user programs, or the batch functions of the executive on a unit basis.

Tasks executed by the demand terminal user normally have frequent but short bursts of computation. Progress is always insisted upon; however, to receive a substantial amount of computation may require a long period of time. Access to computation is a percentage of the total computing facility and is scheduled in small increments of time at frequent intervals to provide immediate responses. This action gives the appearance of total system control to the user and the impression that he is the only user currently running. The more a user is required to interact with a demand program, the shorter the bursts of computation required to service a given request. The bursts of computation are time-shared within the executive to provide an apparent immediate response, with the program placed in a dormant mode during idle periods awaiting response from the user.

While a demand program is in a dormant mode, it may be necessary to swap the program from main storage. Normally, this happens only when main storage is full and another program, which is currently on mass storage, has work to do.

The following is a brief discussion of the operational procedures for initialization of a demand terminal, submission of a demand run, termination of the demand run, and deactivation of the demand terminal.

1.2. Establishing a Connection

1. Dial the UNIVAC 1108.

On a Teletype machine, press the ORIG button, wait for the dial tone, and dial the assigned telephone number. If the number is incorrectly dialed, or a busy signal is received, press the CLR button and try again.

On other types of terminals, follow the manufacturer's instructions.

2. Wait for the data tone.

The data tone is a "beep" which indicates that the data communication circuit between the demand terminal and the 1108 is complete. If the data tone is not received, press CLR and try again.

3. Type in the siteid assigned to the terminal you are using.

Each device that is registered as a demand terminal with Georgia Tech has a unique six-character siteid assigned to it. The siteid should be posted on or near the demand terminal and will be of the form

UGT0xx	110 baud,	72 character line
UFGTxx	110 baud,	132 character line
UGT3xx	110/300 baud,	72 character line
UDGTxx	110/300 baud,	132 character line

where each x represents a digit.

While typing the siteid, no carriage-returns or line-feeds are required. If a typing mistake is made, merely begin again with the first character.

If the siteid is correctly typed but no response is received, re-type several times. If a response is still not received, hold down the CTRL key, press the EOT key (this will normally disconnect you), and redial. If this also fails, the 1108 may be down or the central site operator may have communications disabled. Press CLR and try again later.

4. Receive the sign-on message.

When EXEC 8 receives the siteid, verifies that it is defined and not active, it will respond with the message

UNIVAC 1108 TIME/SHARING EXEC VERS. x

and provide a blacked-out area for the password (see 1.3). The x in the above message represents the version of EXEC 8 in use at the present time, and should be mentioned in reporting troubles.

After the above is accomplished, the user is in communication with an EXEC 8 demand symbiont which controls all input and output to the terminal. The following conventions apply at all times:

a. The carriage-return (CR) indicates the completion of an input line, and the symbiont will indicate acceptance by transmitting a line-feed (LF).

b. The question mark (?) indicates that the current line is to be discarded. The symbiont transmits CR, LF and waits for the next line.

c. The left-arrow (←) (underline on some terminals) indicates that the previous character on the line is to be "erased." As many left-arrows as desired may be used.

1.3. Initiating a Run

1. Type in your password over the blacked-out area, followed by a carriage-return.

A unique, six-character password is assigned to each authorized 1108 demand user. Care should be taken not to disclose the password to anyone.

If a typing mistake is made and detected prior to pressing the carriage return, type a question mark (?), roll the platen back one line, and retype the entire password.

The symbiont will respond with the message NO RUN ACTIVE, indicating that the password has been received. Checking the password is deferred until after a @RUN statement has been entered.

2. You may now enter a @RUN statement as described in Section 2.2 of this manual, followed by your "runstream" (control and data statements).

```
UGT003

UNIVAC 1108 TIME/SHARING EXEC  VERS 27.20.143:30

XXXXXX
NO RUN ACTIVE
@RUN     CCRAY,97C12420,SPALDING-R
DATE: 040372       TIME: 105703
```

Figure 1-1. Establishing a Connection, Initiating a Run

1.4. Terminating a Run

To terminate a demand run, two options are available.

Option 1. Enter a @FIN statement.

The @FIN statement causes all facilities assigned to the run to be released, and prints accounting information on the terminal.

@FIN should not be typed unless the run is in "control mode;" i.e., EXEC 8 is waiting for control statement input. If, for example, @FIN is typed when a program is executing, the symbiont will not allow any input until the program has terminated and all remaining control statements have been processed. This may cause unnecessary waiting by the demand user.

After the accounting information is printed, the symbiont will provide another blacked-out area so that another run may be initiated from the same site. If you want to disconnect instead, follow the disconnect procedure of Section 1.5.

```
@FIN


     RUNID:  CCRAY     REF. NO: 97C12420              NAME: SPALDING-R

     TIME: 00:00:00.019     IN: 2        OUT: 0          PAGES:   1

     INITIATION  TIME:     10:57:03-APR  3, 1972

     TERMINATION TIME:     10:57:42-APR  3, 1972

MMMMMM
```

Figure 1-2. Terminating a Run (Option 1)

Option 2. Hold down the CTRL key, and press the EOT key.

The symbiont will terminate the run and disconnect the terminal. No accounting information is printed.

It should be noted that if a program is active, it will be aborted when EOT is pressed.

1.5. Disconnecting the Terminal

If Option 1 was used to terminate the run, then the terminal may be disconnected by holding down the CTRL key and pressing the EOT key.

Do not use the CLEAR button. This may prevent reestablishing a connection at a later time.

2. EXEC 8 CONTROL STATEMENTS

2.1. Control Statement Format

Control statements from a demand terminal are identified by the master space character (@) in column 1. Every statement that begins with the master space is considered as a control statement. All control statements are "free field" in format (i.e., they are not fixed in format) with a variable number of specifications required. In general, a blank may be replaced by any number of blanks and a comma may be followed by any number of blanks. There are no column restrictions except that the master space must be in column 1.

The general format of a control statement is:

```
@ Command, options    spec1, spec2,...,specn
```

This is the most frequently used form of a control statement. For other forms see GTPRM Sec. 4.2.2.

The 'command' field must always be specified and should be followed by a comma only when 'options' are specified. The other fields are required or optional depending on the control statement used. (See GTPRM chapter 4, "EXECUTIVE CONTROL LANGUAGE").

Examples:

```
@RUN     DEMO,     99X1012,     DOE-JOHN-Q
@RUN DEMO,99X1012,DOE-JOHN-Q
```

(Note: The above two control statements are equivalent.)

```
@XQT
@FOR
@FOR     MAINPROG, MAIN
```

2.2. The @RUN Control Statement

The first control statement that must be entered by the demand user is the @RUN statement. A suitable @RUN statement has the format

```
                @RUN     runid, reference-number, user-name
@RUN,/N  runid, reference-number, user-name, max-time-in-minutes, max-pages
```

The 'runid' field may be from 0 to 6 characters of the user's choice. However, it is recommended that the first two characters of the runid be the same as the first two characters of the user's password. These characters are an abbreviation for the user's school or department, and will ensure that any output produced on the onsite printers or punches will be returned to the proper bin.

The parameters 'reference-number' and 'user-name' will be supplied to the user by his Departmental Computer Coordinator. When an acceptable @RUN statement has been entered, Exec 8 will respond with the time and date as follows:

DATE: mmddyy TIME: hhmmss

meaning that the password and @RUN card are valid (see Figure 1). Further control statements may now be entered. Other possible responses to the @RUN statement are:

1. NO RUN ACTIVE

The input image was not recognizable as a @RUN statement. Run initiation must begin again with step 1, section 1.3. No new black-out area is provided, so the platen should be rolled back.

2. INV. NAME OR REFNO.

The name and/or reference number supplied on the @RUN statement are invalid.

3. INV. PRIORITY REQ.

The priority specified on the @RUN statement is higher than allowed.

4. INV. TIME REQ.

The estimated run time exceeds the maximum allowed for the user.

5. INV. PAGE REQ.

The estimated pages exceed the maximum allowed.

6. INV. ACCESS MODE.

The supplied reference number is not allowed demand access.

7. INCORRECT PASSWORD

The specified password does not belong to the user named on the @RUN statement.

8. BUDGET EXCEEDED

Either the reference number has exhausted its budgeted dollars, or the user has exceeded the charge limit set up for him.

9. BAD RUN STATEMENT

A syntax error was discovered.

In case of responses 2 through 9, a new blacked-out area will be provided. The user may attempt the entire Run Initiation procedure (see 1.3) or follow the disconnect procedure explained in 1.5.

2.3. Commonly Used Control Statements

Four commonly used control statements are presented here. As the user becomes more experienced, he may require others, such as @ASG and "FURPUR" statements. For further information on all control statements, see GTPRM chapters 4 and 5.

1. @FIN

This statement terminates a run which was previously started with a @RUN statement. The user may disconnect the terminal or begin a new run as previously described.

2. @processor

This statement is the processor call statement used to call programs and compilers which are resident on the EXEC 8 system. For example, @FOR is used to call the FORTRAN compiler, ~~@ALG~~ @NUALG for the ALGOL compiler @ACOB for the ANSI COBOL compiler, and @ED for the ED processor.

3. @XQT

This statement is used to execute programs previously compiled by the user, or to execute the application programs provided by EXEC 8.

4. @EOF

This statement may be used to terminate program language statements to a compiler, statements to a processor, or data images to a user program.

3. CREATING AND USING A FILE IN A RUN

This chapter is presented to exemplify demand usage of primarily batch system components, such as FORTRAN V. Although a demand runstream may actually be the same as a batch deck setup, this is not efficient or even convenient.

If the user intends to use demand components of the system, such as BASIC or APL, this chapter may be skipped. Such components usually provide their own, terminal-oriented file system. Such file systems will not be discussed in this manual.

3.1. EXEC 8 Program Files

The EXEC 8 language processors (e.g., the FORTRAN, COBOL, and ALGOL compilers) operate on elements of program files. A program file, by definition, must exist on a random access device. Elements of a program file are of three general types: symbolic, relocatable, and absolute. Symbolic elements contain source programs (e.g., FORTRAN or ANSI COBOL statements) and are used as input to the language processors. The language processors place their output in relocatable elements. The system processor MAP collects one or more relocatable elements into a single absolute element which may be executed with the @XQT control statement.

In simplest form, both program files and elements are named by one to twelve characters from the set A through Z, 0 through 9, -, and $. The character $ is used in all system-defined names and hence should be avoided to prevent conflicts. In the control language, an element named E in a program file named PF is denoted

```
PF.E
```

A program file may contain many elements, provided only that all elements of the same type have unique names. The control statement

```
@PRT, T      PRFIL.
```

will print a listing of the elements in program file PRFIL. This is frequently called a "table of contents," or TOC.

In addition to the element names, the element type will be listed. As mentioned above, the type may be symbolic (SYM), relocatable (REL), or absolute (ABS). Symbolic elements are further classified depending on the source language contained in the element, and thus symbolic elements may be denoted with one of the types:

SYM	general symbolic
ELT	data or control stream
ASM	1100 assembler source language
COB	COBOL source language
FOR	FORTRAN source language
ALG	ALGOL source language
MAP	1100 Collector source statements
DOC	Document processor element
SEC	1100 file administrator source statements

Also BAS, SNO, LISP, etc.

Although source language procedures will not be discussed in this manual, the following symbolic element codes are mentioned for completeness:

ASMP	1100 Assembler procedure
COBP	COBOL Library procedure
FORP	FORTRAN Include procedure

EXEC 8 automatically assigns a temporary program file to every run, and its name is TPF$. In the control language, if a program file name is not specified where one is expected, TPF$ is implied by default. Thus, element MAIN in program file TPF$ may be denoted either

TPF$.MAIN or simply MAIN

In order to run a program, then, a user must

1. Establish a program file.
2. Insert symbolic elements into the program file.
3. Direct language processors to compile the symbolic elements, producing relocatable elements.
4. Make corrections to the symbolic elements, and recompile.
5. Direct the MAP processor to create an absolute element.
6. Direct EXEC 8 to load and execute the absolute element.

The next 6 sections of this chapter describe, primarily by means of examples, how these six steps would typically be accomplished. For more detailed information on program files, see GTPRM Sec. 1.7.

3.2. Establishing a Program File

The control statement

```
@CAT,P     X,F2
```

will catalogue a file named X; that is, it is entered into the file directory. When the system responds with the message, READY, X may be used as a program file.

```
@CAT,P X,F2
READY
@PRT,T X.
FURPUR 023A-04/03-11:16

SPALDING-R*X        IS EMPTY
END PRT
```

Figure 3-1. Establishing a Program File; Using @PRT,T

File names are qualified by the user name from the @RUN statement; thus every user may have a file named X (or anything else), and no interference between users will occur.

A file created in this manner is said to be catalogued. What this means is that when the user terminates his run, the information in the file is saved. The user may then initiate a new run at a later time and retrieve his information.

In the case of program files, this allows the user to conveniently develop and check out a program over an extended period, rather than having to do everything in one session at the terminal. More will be said about this in section 3.8, "Referencing Previously Catalogued Files."

3.3. Inserting a Symbolic Element into a Program File

The system processor, @ED, is a convenient means by which a demand user may insert (and correct) a symbolic element. @ED is called to insert an element by a statement such as

```
@ED,I      X.MAIN
```

The ED processor will solicit images (with consecutive line numbers) to be inserted into program file X, element MAIN. If a symbolic element named MAIN already exists in program file X, it is automatically deleted.

After typing in the element, a carriage return switches ED to EDIT mode so that, if desired, the symbolic type may be set and the element displayed. @EOF (or any control statement) terminates ED.

The @PRT,T statement may be used at this time to see that program file X actually contains symbolic element MAIN. (See Figure 3-2).

3.4. Compilation

3.4.1. General

After a user has placed source statements in a symbolic element of a program file, he should direct the appropriate compiler to process it. This is done by means of the language processor call statement, a suitable format of which is:

```
@processor, options   si-eltname, ro-eltname
```

'processor' names the compiler to be used. Often used compilers include:

FOR	FORTRAN V (@UFOR for FORTRAN V, @FOR or @RFOR for FORTRAN IV)
ALG	ALGOL (@NUALG)
ACOB	ANSI COBOL

'options' define the listing desired, the handling of symbolic elements, and processor-dependent features.

'si-eltname' names the program file and element to be used as source input.

'ro-eltname' names the program file and element in which the relocatable output will be placed. Usually, this may be left blank in which case it is assumed to be the same as 'si-eltname'.

An @EOF statement (or other control statement) must be entered following the language processor call statement before compilation will be completed.

```
@ED,I X.MAIN
ED 12.02-04/03-11:17-(,0)
INPUT
1I:         SUM=0.
2I:         N=0
3I:         WRITE(6,191←←01)
4I:101      FORMAT(' ENTER NUMBERS TO BE AVERAGED')
5I:5        READ(5,102,END=10)VAL
6I:102      FORMAT(E20.5)
7I:         SUM=SUM+VAL
8I:         N=N+1
9I:         GO TO 50
10I:10       AVG=SUM/N
11I:         WRITE(6,103)SUM,N,AVG
12I:103      FORMAT(' SUM=',E10.3,' N=',I10,' AVG=',E10.3
13I:         STOP
14I:         END
15I:
EDIT
14:TYPE 4←FOR
14:PRINT!
       SUM=0.
       N=0
       WRITE(6,101)
101    FORMAT(' ENTER NUMBERS TO BE AVERAGED')
5      READ(5,102,END=10)VAL
102    FORMAT(E20.5)
       SUM=SUM+VAL
       N=N+1
       GO TO 50
10     AVG=SUM/N
       WRITE(6,103)SUM,N,AVG
103    FORMAT(' SUM=',E10.3,' N=',I10,' AVG=',E10.3
       STOP
       END
SCAN:14
EOF:14
0:@EOF
LINES:14
@PRT,T X.
FURPUR 023A-04/03-11:21

SPALDING-R*X
FOR  MAIN(1)
END PRT
```

Figure 3-2. Using ED to Insert an Element

3.4.2. FORTRAN V

Don't use any option at all.

For FORTRAN V compilations, the D option is recommended for demand runs. This will list only the lines with errors along with the error messages. However, the S option may be used to list the entire program. Complete details on options, except for the D option, are contained in the UNIVAC FORTRAN V Programmer's Reference Manual.

3.4.3. ANSI COBOL

Try BEST options.

Recommended options for ANSI COBOL are BE. This will list errors and inhibit sequence checking.

When using ANSI COBOL from a demand terminal, the compiler will request information regarding the margins used in the source language. This is done to allow free-form input from the terminal. The user must follow the instructions printed, and answer A, B, or C.

The options BES may be used for a listing of the whole program.

3.4.4. ALGOL

See Hardware/Software Summary under NUALG

There is currently no option to list only errors. The user must list the whole program (S option) or none of it (N option). Also, the 'ro-eltname' field must be specified. These restrictions will be removed in the future.

```
@FOR,D X.MAIN
FOR S9A-04/03/72-11:24:43 (0,)
@EOF

00115        *ERROR*  THERE IS NO LABEL 50.
00115       9*             GO TO 50
00115        *ERROR*  THIS ASSIGNMENT IS MEANINGLESS SINCE THERE IS NO PO
      SSIBLE REFERENCE TO THIS LABEL.
00124  *DIAGNOSTIC*  THIS STATEMENT HAS TOO MANY LEFT PARENTHESES.
00124      12*      103   FORMAT(' SUM=',E10.3,' N=',I10,' AVG=',E10.3

        END OF COMPILATION:          3 DIAGNOSTICS.
         ERRORS  IN  RELOCATABLE  ELEMENT
@PRT,T X.
FURPUR 023A-04/03-11:25

SPALDING-R*X
FOR  MAIN(1)
REL  MAIN
END PRT
```

Figure 3-3. Compiling an Element Using FOR.

3.5. Correcting Symbolic Elements

The ED processor can be used to correct and update symbolic elements. The appropriate control statement to call ED to update element MAIN in program file X is

```
@ED,U     X.MAIN
```

The details of using ED are covered in Chapter 4 of this manual. However, a few widely used capabilities of ED will be described below.

Initially, ED copies the element to a scratch file. A pointer is set to line 0, ahead of line 1 (the actual first line). The element is processed image by image, from top to bottom, and then the pointer is reset to line 0.

ED solicits editing commands by displaying the line number of the image currently pointed to. The user may change the pointer by typing an unsigned number (which moves the pointer to that line), a plus sign and a number (which moves forward the specified number of images), or a minus sign and a number (which moves backward). In addition, the user may search for a particular character combination by using the LOCATE (L) and FIND (F) commands.

Suppose it is desired to change the line GO TO 50 to GO TO 5. The pointer must be advanced to that line. If it is the first line on which the string GO appears,

```
L GO
```

will locate the proper line. ED types out the image when found, then displays its number and waits for other commands.

The CHANGE (C) command may be used to alter the image presently pointed to. The format is:

```
C   delim   string1   delim   string2   delim
```

Any single, non-blank character may be used as the delimiter, 'delim', as long as it does not appear in 'string1' or 'string2'.

Thus, to change the GO TO 50 to GO TO 5, the command

```
C /50/5/
```

may be used. ED will print the result and wait for further commands.

The FIND (F) command is similar to the locate command, except that fixed columns are used. In FIND 103, a match occurs only when 103 is in columns 1 through 3 of an image.

Any control statement indicates that all editing on the element is complete. ED will copy its scratch file back to the program file specified on the @ED statement, file a new symbolic element of the name specified, and delete the old symbolic element.

Of course, after a symbolic element has been updated, it must be recompiled as in Section 3.4.

```
@ED,U X.MAIN
ED 12.02-04/03-11:27-(0,1)
EDIT
0:L GO
      GO TO 50
9:C /50/5/
      GO TO 5
9:FIND 101
EOF:14
0:FN-IND 103
103   FORMAT(' SUM=',E10.3,' N=',I10,' AVG=',E10.3
12:C /G=',E10.3/G=',E10.3)/
103   FORMAT(' SUM=',E10.3,' N=',I10,' AVG=',E10.3)
12:@FOR,D X.MAIN
LINES:14
FOR S9A-04/03/72-11:29:13 (1,)
@EOF

        END OF COMPILATION:          NO  DIAGNOSTICS.
```

Figure 3-4. Updating and Re-compiling an Element

3.6. Creating an Absolute Element

The MAP processor (also called the COLLECTOR) is used to combine one or more relocatable elements into a single absolute element.

3.6.1. MAP Processor Call

A suitable MAP processor call statement is:

```
@MAP,IN     ,X.MAIN
```

Put the name you desire for your program here.

where:

I option	- specifies that MAP directives follow in the input stream
N option	- specifies that no listing (other than errors) is to be produced
spaces before the second comma	- specify that the MAP directives are not to be saved in a symbolic element
X.MAIN	- specifies that the resulting absolute element is to be called MAIN and placed in program file X.

3.6.2. MAP Directives

The IN directive is used to specify relocatable elements that are to be included in the resulting absolute element. Some examples are:

```
IN   X.MAIN
```

Include relocatable element MAIN from program file X.

```
IN   X.MAIN, X.SUB1, X.SUB2
```

Include relocatable elements MAIN, SUB1, and SUB2 from program file X.

```
IN   X.MAIN, MYNE.CALLER
IN   MYNE.XYZ
IN   X.SUB1
```

Include relocatable elements MAIN and SUB1 from program file X, and relocatable elements CALLER and XYZ from program file MYNE.

```
IN   X.
```

Include all relocatable elements in program file X.

```
IN   X
```

Include relocatable element X from program file TPF$. (Note that the period is quite significant here!)

The LIB directive is used to specify program files to be used as library files. Relocatable elements in a library file will be included in the absolute element only if they are referenced by another element already selected for inclusion. In order to be used as a library file, a program file must have been prepared by the FURPUR @PREP command. (See GTPRM, 5.4.8) Example:

```
LIB   USERLIB.
```

The computer center maintains several standard library files that users may specify on a LIB directive. Such files are @PREP'ed by the computer center. For example, the MATH-PAK and STAT-PAK routines are kept in the program file

```
SYSTEM$*MATHSTAT.
```

Thus, if the user has a relocatable X.MAIN that references one or more MATH-PAK routines, appropriate MAP directives would be:

```
IN   X.MAIN
LIB  SYSTEM$*MATHSTAT
```

NOTE: SYSTEM$ is called the qualifier, and is separated from the file by the *. If the qualifier were not specified, the user name would be used by default (see Section 3.2.), and the proper file would not be used.

Additional directives are described in the GTPRM, Section 6.1.3.

The user must indicate the end of directives with an @EOF statement, or other control statement.

3.6.3. System Relocatable Library

The collector will automatically include referenced relocatable elements from the system relocatable library, SYS$*RLIB$. Routines in this file include standard math routines (SQRT, LOG, SIN, etc.), input/output routines (READ, WRITE, etc.), and various other routines that perform utility functions for the user program.

3.6.4. ANSI COBOL MAP

All users of @ACOB (ANSI COBOL) should use the following MAP directive when they collect their absolute:

```
LIB   SYSTEM$*EXTRA
```

ANSI COBOL users should also note that the compiler generates several relocatable elements for one symbolic. Therefore, IN directives may be more complicated than for FORTRAN or ALGOL programs. Consult the ANSI COBOL Programmers Reference Manual, and see the example in Section 3.10.2.

3.7. Program Execution

The control statement

```
@XQT   X.MAIN
```

will cause EXEC 8 to load and transfer control to absolute element MAIN in program file X.

```
@MAP,IN   ,X.MAIN
MAP 0023-04/03-12:58

IN X.MAIN
@EOF


@PRT,T X.
FURPUR 023A-04/03-12:58

SPALDING-R*X
FOR  MAIN(2)
REL  MAIN
ABS  MAIN
END PRT
```

Figure 3-5. Creating an Absolute Element

If the program being executed is set up to read data images from the "card reader," such images are entered immediately following the @XQT control statement. If the program directs output to the "printer," such output will be displayed on the terminal. In FORTRAN, the card reader is unit 5, and the printer is unit 6. In ALGOL, the assumed input device is the card reader and the assumed output device is the printer; hence if no devices are specified in READ and/or WRITE procedure calls, the input and output will automatically be directed to the terminal. In ANSI COBOL, files for which terminal input/output is desired would be ASSIGNed to PRINTER and CARD-READER in the ENVIRONMENT DIVISION.

Program execution can usually be terminated by entering the control statement

```
@EOF
```

(or, in fact, any control statement) which will cause an end-of-file condition to be returned to the program when it attempts to read the statement as data. If a program cannot be terminated in this way, the BREAK key on the teletype may be used as described in Section 6.3.1.4.

@@X TIO will kill anything but CTS, painlessly.

```
@XQT X.MAIN
 ENTER NUMBERS TO BE AVERAGED
5.0
10.0
15.0
@EOF
 SUM=    .300+02 N=          3 AVG=    .100+02
END        54 MLSEC
```

Figure 3-6. Execution of a Program

The program may be executed again without modification simply by entering another @XQT control statement.

```
@XQT X.MAIN
 ENTER NUMBERS TO BE AVERAGED
19.70
31.001
16.35
128.37
29.47
@EOF
 SUM=    .225+03 N=          5 AVG=    .450+02
END        51 MLSEC
```

Figure 3-7. Re-execution of the Same Program

If it is desired to change the program following execution, the @ED,U statement may be used as in Section 3.5. If a symbolic element is changed, it should be recompiled. The MAP processor should be called again following all recompilations. It is not necessary to recompile symbolic elements which were not changed.

3.8. Referencing Previously Catalogued Files

3.8.1. General

If the user terminates his run, and then initiates a new run at a later time, he may retreive his programs that were placed in catalogued

files. This may be done to update the programs or simply execute them with new data.

In all cases, the file(s) in question should be assigned as follows:

```
@ASG,A    X
```

@ASG,AZ X will keep you from having a "WAITING ON FACILITY" message

'X', of course, is the name of the file.

3.8.2. System Responses to @ASG,A

The most common responses are covered below. A complete set of responses is given in the GTPRM, Section 4.5.2.6.

READY

The assign was completed normally.

FACILITY WARNING 000000000200

The file was assigned when the system failed (see Chapter 5). This is a warning only, and means that the file may have been only partially updated by an updating run. The user should examine the file (with, say, a @PRT,T), and after noting its condition, enter an @ENABLE filename command.

FACILITY REJECTED 400000000100

The file was not properly reloaded from a computer center backup tape. An @ENABLE filename should be done, followed by another @ASG,A. If this fails, @DELETE the file and recreate it. If recreation would be difficult, contact a Programmer's Aide as he may be able to load an older version.

FACILITY REJECTED 400000000400

The file was destroyed by a hardware or software error. @ENABLE may be attempted, but it is unlikely that the file will be intact. @DELETE it, then recreate it or contact a Programmer's Aide.

FACILITY REJECTED 400010000000

The file is not catalogued! If spelling is correct, then the computer center may have deleted the file while enforcing its mass storage policy. The following classes of files are likely to be deleted:

(1) catalogued with equipment type other than F2. (or F40 or F44) This includes files catalogued with the 'F2' omitted from the @CAT statement.

(2) of zero size. If a file has nothing in it, it is a waste of resources to maintain it.

(3) of size greater than the "maximum allowable size."

(4) files that have not been assigned for a period longer than the "expiration period."

A Programmer Aide may be able to reload the file if it cannot be conveniently recreated by the user.

Additional classes of files may be designated to be deleted in the future. Your Departmental Computer Coordinator will have current information concerning mass storage policy, including the current "maximum allowable size" and "expiration period."

WAITING ON FACILITY

This may mean two different things: someone else is using the file, or the file is "unloaded." In either case, the run will be held until the file is available; at that time, the message READY will be displayed. If the user wishes to do something else while waiting for availability, the BREAK key may be used as described in Section 6.3.1.4 to remove the hold condition. If the BREAK key is used, the assign must be reattempted later before the file may be used. @@X TO instead of Break or Message Waiting keys.

```
@ASG,A X
READY
```

Figure 3-8. @ASG, A Statement and Normal System Response

3.9. Catalogued File Maintenance

From time to time, the user should perform maintenance (housekeeping) on his files. This includes deleting files no longer needed, deleting elements no longer needed, and "packing" program files to release space occupied by deleted elements.

3.9.1. The List Files Processor

The LF (list files) processor may be used to list all the files belonging to the user. Complete details are in GTPRM Section 6.7; commonly used features are described below.

@LF Slow, slow, slow! @PRT,P is faster, but still slow.

will list all files belonging to the user.

@LF, options

will list certain information about all the user's files. Commonly used options are:

L - list everything.

S - list the size in tracks (a track contains 1792 36-bit words).

T - list the time and date of cataloguing and of last assign.

P - list the cataloguing parameters (specification 2 of the @CAT).

O - list the @CAT options.

B - list computer center backup information: TFW means the time that the file was first written into (if such was done) following the last time the file was copied to tape. BU means the time the file was last copied to tape.

@LF, options filename, ...

will list the requested information about the specified files only.

Certain files may be annotated as follows:

-UNLOADED- The text of the file exists only on tape. If the user assigns the file, he will be placed in WAITING ON FACILITIES mode while the text is being reloaded.

-DISABLED- -INCOMPLETE WRITE- The file was assigned during a system failure. See Section 3.8.2.

-DISABLED- -DESTROYED- A hardware or software error destroyed the file. See Section 3.8.2.

-DISABLED- -BACKUP CANNOT BE READ OR LOAD IN PROGRESS- An assign will receive FACILITY REJECTED 400000000100. See Section 3.8.2.

The user should periodically list the names of all his files, and delete those no longer needed.

3.9.2. The @DELETE Statement

The @DELETE statement may be used to delete files or delete elements of program files.

```
@DELETE   filename
```

deletes the specified file.

```
@DELETE,options filename.eltname
```

deletes element 'eltname' from program file 'filename'. Since 'eltname' may exist concurrently as symbolic, relocatable, and absolute, the 'options' indicate the type(s) of elements to delete, as follows:

A option - absolute element
R option - relocatable element
S option - symbolic element

The user should periodically delete files and elements no longer needed.

3.9.3. The @PACK Statement

The @PACK statement rewrites an entire program file so as to exclude deleted elements and release unneeded mass storage. Remember that in addition to elements explicitly deleted by the @DELETE statement, elements are deleted each time an element is updated, recompiled, remapped, or re-inserted. It is important that users pack their program files frequently, for the computer center typically finds thousands of mass storage tracks tied up in deleted elements. In addition, failure to pack frequently may cause a file to exceed its maximum size prematurely.

This includes BASIC programs.

The format of the @PACK statement is

```
@PACK filename
```

or

```
@PACK filename, filename, filename  (etc.)
```

(As many files as desired may be specified.)

Neither BREAK nor EOT should be pressed when a pack is being done, as this will destroy the file. Always wait for the END PACK message.

```
@PRT,T X.

SPALDING-R*X
FOR  MAIN(2)
REL  MAIN
ABS  MAIN
END PRT
@DELETE,R X.MAIN
END DELETE
@PRT,T X.

SPALDING-R*X
FOR  MAIN(2)
ABS  MAIN
END PRT
@LF,S X.
LF-12-04/03/72-13:01:24
SPALDING-R*X(1)                    SZ-7
END LF  7 TRACKS
@PACK X.
FURPUR 023A-04/03-13:02
END PACK
@LF,S X.
LF-12-04/03/72-13:02:23
SPALDING-R*X(1)                    SZ-4
END LF  4 TRACKS
```

@PRT,TL <filename> will give a very detailed listing of what's in a file, and will also show you how many elements are deleted.

Note size difference.

Figure 3-9. Catalogued File Maintenance

3.10. Examples

The examples in this section are reprinted from actual demand terminal output. Having previously studied Chapter 3, the reader should understand these examples except for a few techniques; these are pointed out in the text included with each example.

3.10.1. A Complete ALGOL Run with Debugging

This example serves two purposes. First, it shows the ALGOL programmers how to apply the techniques explained previously in this chapter for FORTRAN. Second, it provides a review of the procedures used for any language. For the latter reason, all users would benefit from reading this example.

Note that the standard ED TAB character is the semicolon which is used in the source language. Therefore, the TAB feature is inhibited before the program is entered. An alternative would be to use the dollar sign in the source language rather than a semicolon.

NUALG does not accept dollar signs. Use a ;

```
UGT003

UNIVAC 1108 TIME/SHARING EXEC  VERS 27.20.143:32

[illegible]
NO RUN ACTIVE
@RUN    CCRAY,97C12420,SPALDING-R
DATE: 041372       TIME: 113507
@ASG,A X
READY
@ED,I X.ALGTEST
ED 12.02-04/13-11:36-(,0)
INPUT
1I:
EDIT
0:TAB
0:TYPE ALG
0:
INPUT
1I:BEGIN
2I:INTEGER N; REAL VAL,SUM;
3I:FORMAT F103 ('SUM=',R10.3,' N=',I10,'AVG=',R10.3);
4I:LOCAL LABEL EOFLAB;
5I:WRITE('ENTER NUMBERS \TO BE AVERAGED');
6I:READONE: READ(VAL,EOFLAB);
7I:SUM=SUM+VAL
8I:N=N+1
9I:GO TO READONE;
10I:EOFLAB:AVG=VAL/N;
11I:WRITE(SUM,N,AVG,F103) END
12I:
EDIT
11:LNPRINT!
    1:BEGIN
    2:INTEGER N; REAL VAL,SUM;
    3:FORMAT F103 ('SUM=',R10.3,' N=',I10,'AVG=',R10.3);
    4:LOCAL LABEL EOFLAB;
    5:WRITE('ENTER NUMBERS TO BE AVERAGED');
    6:READONE: READ(VAL,EOFLAB);
    7:SUM=SUM+VAL
    8:N=N+1
    9:GO TO READONE;
   10:EOFLAB:AVG=VAL/N;
   11:WRITE(SUM,N,AVG,F103) END
SCAN:11
EOF:11
0:
```

Not a good sample of ALGOL, does not show how simple it is to outdo FORTRAN, especially FOR statements and PROCEDURE calls.

Figure 3-10. ALGOL Example (Page 1 of 4)

```
0:7
SUM=SUM+VAL
7:C /VAL/L-VAL;/
SUM=SUM+VAL;
7:+1
N=N+1
8:C /1/1;/
N=N+1;
8:EXIT
LINES:11
@ALG X.ALGTEST,X.ALGTEST
CYCLE  000  COMPILED BY  1204  0008  ON 04/13/72 AT 11:41:36
@EOF
@MAP,IN ,X.ALGTEST
MAP 0023-04/13-11:41

IN X.ALGTEST
@EOF


ERROR IN ELEMENT: ALGTEST
@ALG,S X.ALGTEST,X.ALGTEST
CYCLE  000  COMPILED BY  1204  0008  ON 04/13/72 AT 11:42:19
@EOF
      B1
     1        BEGIN
      S1      L1
     2        INTEGER N; REAL VAL,SUM;
     3        FORMAT F103 ('SUM=',R10.3,' N=',I10,'AVG=',R10.3);
     4        LOCAL LABEL EOFLAB;
     5        WRITE('ENTER NUMBERS TO BE AVERAGED');
     6        READONE: READ(VAL,EOFLAB);
     7        SUM=SUM+VAL;
     8        N=N+1;
     9        GO TO READONE;
    10        EOFLAB:AVG=VAL/N;
                        *
UNDEFINED VARIABLE
      E1
    11        WRITE(SUM,N,AVG,F103) END

  1 ERROR(S) WERE FOUND
@ED,U X.ALGTEST
ED 12.02-04/13-11:43-(0,1)
EDIT
0:2
INTEGER N; REAL VAL,SUM;
2:C /SUM/AVG,A-SUM/
INTEGER N; REAL VAL,AVG,SUM;
2:3
FORMAT F103 ('SUM=',R10.3,' N=',I10,'AVG=',R10.3);
3:C /AVG/ AVG/
FORMAT F103 ('SUM=',R10.3,' N=',I10,' AVG=',R10.3);
3:EXIT
LINES:11
```

Figure 3-10. ALGOL Example (Page 2 of 4)

```
@ALG,N X.ALGTEST,X.ALGTEST
CYCLE  001  COMPILED BY  1204  0008  ON 04/13/72 AT 11:44:54
@EEF--OF
@MAP,IN X-?
@MAP,IN ,X.ALGTEST
MAP 0023-04/13-11:45

IN X.ALGTEST
@EOF


@XQT X.ALGTEST
ENTER NUMBERS TO BE AVERAGED
5.0
10.0
15.0
@EOF
END       52 MLSEC
@ED,U X.ALGTEST
ED 12.02-04/13-11:46¬(1,2)
EDIT
0:L FORMAT
FORMAT F103 ('SUM=',R10.3,' N=',I10,' AVG=',R10.3);
3:C /)/,A1)/
FORMAT F103 ('SUM=',R10.3,' N=',I10,' AVG=',R10.3,A1);
3:@ALG X.ALGTEST,X.ALGTEST
LINES:11
CYCLE  002  COMPILED BY  1204  0008  ON 04/13/72 AT 11:46:49
@EOF
@MAP,IN  ,X.ALGTS-EST
MAP 0023-04/13-11:47

IN X.ALGTEST
@EOF


@XQT X.ALGTEST
ENTER NUMBERS TO BE AVERAGED
5.0
10.0
15.0
@EOF
SUM=  3.00,+01 N=          3 AVG=  5.00,+00
END       58 MLSEC
@ED,U X.ALGTEST
ED 12.02-04/13-11:48-(2,3)
EDIT
0:F EOFLAB
EOFLAB:AVG=VAL/N;
10:C //-VAL/SUM/
EOFLAB:AVG=SUM/N;
10:@ALG X.ALGTET-ST,X.ALGTEST
LINES:11
CYCLE  003  COMPILED BY  1204  0008  ON 04/13/72 AT 11:49:40
@EOF
```

Figure 3-10. ALGOL Example (Page 3 of 4)

```
@MAP,IN ,X.ALGTEST
MAP 0023-04/13-11:50

IN X.ALGTEST
@EOF

@XQT X.ALGTEST
ENTER NUMBERS TO BE AVERAGED
5.0
10.0
15.0
@EOF
SUM=  3.00,+01 N=          3 AVG=  1.00,+01
END       60 MLSEC
@XQT X.ALGTEST
ENTER NUMBERS TO BE AVERAGED
19.70
31.001
16.35
128.37
29.47
@EOF
SUM=  2.25,+02 N=          5 AVG=  4.50,+01
END       70 MLSEC
@PACK X
FURPUR 023A-04/13-11:52
END PACK
@PRT,T X.

SPALDING-R*X
FOR  MAIN(2)
ABS  MAIN
ALG  ALGTEST(4)
REL  ALGTEST
ABS  ALGTEST
END PRT
@FIN


    RUNID:  CCRAY    REF. NO: 97C12420          NAME: SPALDING-R

    TIME: 00:00:06.741    IN: 87    OUT: 0        PAGES:  3

    INITIATION  TIME:    11:35:07-APR 13,1972

    TERMINATION TIME:    11:53:06-APR 13,1972

*******
```

Figure 3-10. ALGOL Example (Page 4 of 4)

Note that this user properly PACK'ed his file before terminating. After the blacked-out area was printed, he held down the CTRL key and pressed the EOT key.

3.10.2. An ANSI COBOL Program

This example shows the procedures used to debug and run ANSI COBOL programs. The symbolic element X.COBTEST was previously entered using ED.

Interesting points in this example include the @PRT,T following compilation that shows that ACOB generates several relocatables, and the @PREP and @MAP which are unique for ANSI COBOL programs.

Beware of the absurdity that follows . . . .

```
@ED,R X.COBTEST
READ-ONLY MODE
ED 12.02-04/14-13:25-(6,)
EDIT
0:LNPRINT!
    1:        IDENTIFICATION DIVISION.
    2:        PROGRAM-ID. EXAMPLE.
    3:        ENVIRONMENT DIVISION.
    4:        CONFIGURATION SECTION.
    5:        SOURCE-COMPUTER. UNIVAC-1108.
    6:        OBJECT-COMPUTER. UNIVAC-1108.
    7:        INPUT-OUTPUT SECTION.
    8:        FILE-CONTROL.
    9:            SELECT INFILE ASSIGN TO CARD-READER.
   10:            SELECT OUTFILE ASSIGN TO PRINTER.
   11:        DATA DIVISION.
   12:        FILE SECTION.
   13:        FD  INFILE, LABEL RECORDS STANDARD.
   14:        01  CARD.
   15:            02 DVAL PIC 9999V999.
   16:            02 FILLER PIC X(72).
   17:        FD  OUTFILE, LABEL RECORDS STANDARD.
   18:        01  ANSLINE.
   19:            02 DSUM PIX ZZZ9.999.
   20:            02 FILLER PIC XX.
   21:            02 DNUM PIC ZZZZZZZ9.
   22:            02 FILLER PIC XX.
   23:            02 DAVG PIC ZZZ9.999.
   24:        01  HDR PIC X(132).
   25:        WORKING-STORAGE SECTION.
   26:        77  SUM PIC 99999V999 VALUE 0.
   27:        77  NUM PIC 9999 VALUE 0.
   28:        77  AVG PIC 99999V999 .
   29:        PROCEDURE DIVISION.
   30:        OPEN-UP.  OPEN INPUT INFILE, OUTPUT OUTFILE.
   31:            MOVE 'ENTER NUMBERS TO BE AVERAGED' TO HDR.
   32:            WRITE HDR.
   33:            MOVE '9999V999 IS THE PICTURE' TO HDR WRITE HDR.
   34:        GETVAL.
   35:            READ INFILE AT END GO TO GETAVG.
   36:            ADD 1 TO NUM.
   37:            ADD DVAL TO SUM.
   38:            GO TO GETVAL.
   39:        GETAVG.
   40:            CLOSE INFILE.
   41:            DIVIDE SUM BY NUM GIVING AVG.
   42:            MOVE '      SUM         NUM         AVG' TO HDR
   43:            WRITE HDR.
   44:            MOVE SPACES TO ANSLINE.
   45:            MOVE SUM TO DSUM.  MOVE NUM TO DNUM.  MOVE AVG TO DAVG.
   46:            WRITE ANSLINE.
   47:        CLOSE-UP.
   48:            CLOSE OUTFILE.
   49:            STOP RUN.
SCAN:49
EOF:49
0:@EOF
NO CORRECTIONS APPLIED.
```

Figure 3-11. ANSI COBOL Example (Page 1 of 3)

```
@ACOB,BE X.COBTEST
COB00A5-04/14-13:57-(6,)

THE A AND B MARGINS WILL BE COL. 1 AND 2,(NO SEQ. NUMBERS)
CONTINUATION MARK MUST BE IN COL. 1, TEXT OF CONTIN IN COL 2
ABOVE FORMAT APPLIES TO..A-NONE,B-TTY,C-TTY+PF INPUTS
PLEASE TYPE A B OR C,THEN CARR. RET.A
@EOF
ERROR*   0019      3      UNRECOGNIZABLE OR MISPLACED WORD.
ERROR    0019      4      CHARACTER AFTER PUNCTUATION IS BEGINNING OF NEXT
       WORD.
ERROR*   0019      4      UNRECOGNIZABLE OR MISPLACED WORD.
ERROR*   0019             CLASS IS UNSPECIFIED FOR ELEMENTARY ITEM.
ERROR*   0019             SIZE OF ITEM IS MISSING.
ERROR    0046            NUMERIC MOVE MAY RESULT IN LEFT TRUNCATION
ERROR    0045             ILLEGAL MOVE - NUMERIC TO ALPHA, SPECIAL, OR NON
      -NUMERIC

ERROR* INDICATES FATAL ERROR.


END  COB  ERRORS:       4

COMPILE TIME IS  0000.93 SECONDS
@ED,U X.COBTEST
ED 12.02-04/14-13:59-(6,7)
EDIT
0:19
           02 DSUM PIX ZZZ9.999.
19:C /PIX/PIC/
           02 DSUM PIC ZZZ9.999.
19:45
          MOVE SUM TO DSUM.  MOVE NUM TO DNUM.  MOVE AVG TO DAVG.
45:@ACOB,BE X.COBTEST
LINES:49
COB00A5-04/14-14:00-(7,)

THE A AND B MARGINS WILL BE COL. 1 AND 2,(NO SEQ. NUMBERS)
CONTINUATION MARK MUST BE IN COL. 1, TEXT OF CONTIN IN COL 2
ABOVE FORMAT APPLIES TO..A-NONE,B-TTY,C-TTY+PF INPUTS
PLEASE TYPE A B OR C,THEN CARR. RET.A
@EOF
ERROR    0046            NUMERIC MOVE MAY RESULT IN LEFT TRUNCATION
ERROR    0045            NUMERIC MOVE MAY RESULT IN LEFT TRUNCATION


END  COB  ERRORS:       0

COMPILE TIME IS  0000.95 SECONDS
```

Figure 3-11. ANSI COBOL Example (Page 2 of 3)

```
@PRT,T X.
FURPUR 023A-04/14-14:02

SPALDING-R*X
FOR  MAIN(2)
ABS  MAIN
ALG  ALGTEST(4)
REL  ALGTEST
ABS  ALGTEST
COB  COBTEST(5)
REL  INFILE
REL  OUTFILE
REL  CSWORKCOBTES
REL  C$0101COBTES
REL  C$0102COBTES
REL  COBTEST
END PRT
@PREP X.
END PREP
@MAP,IN ,X.COBTEST
MAP 0023-04/14-14:03

IN X.COBTEST
LIB X,SYSTEM$*EXTRA
@EOF


@XQT X.COBTEST
ENTER NUMBERS TO BE AVERAGED
9999V999 IS THE PICTURE
0005000
0010000
0015000
@EOF
      SUM         NUM         AVG
   30.000           3      10.000
@XQT X.COBTEST
ENTER NUMBERS TO BE AVERAGED
9999V999 IS THE PICTURE
001970
0031001
0016350
0128370
0029470
@EOF
      SUM         NUM         AVG
  224.902           5      44.980
```

Figure 3-11. ANSI COBOL Example (Page 3 of 3)

3.10.3. FORTRAN With Subprograms

In this example, a main program, function, and subroutine are entered into separate symbolic elements, separately compiled, and collected together with the MAP processor.

Of course, you can do this with any language, and in most cases mix languages.

E.G., ALGOL and FORTRAN!

```
BEGIN
   EXTERNAL FORTRAN PROCEDURE OUTPUT;
   INTEGER I; REAL X;
   READ(I);
   FOR X:=(1.0,1.0,I) DO OUTPUT(SQRT(X))
END
```

```
      SUBROUTINE OUTPUT(X)
      PRINT 5,X
5     FORMAT( )
      RETURN
      END
```

Note fortran free-format.

```
@ED,I X.CORMAIN
ED 12.02-04/17-15:25-(,0)
INPUT
1I:
EDIT
0:TYPE FOR
0:SET 7
0:
INPUT
1I:;WRITE(6,100)
2I:100;FORMAT(' ENTER DATA:')
3I:2;READ(5,101,END=900,ERR=800)X,Y
4I:101;FORMAT()
5I:;N=N+1
6I:;SUMX=SUMX+X
7I:;SUMY=SUMY+Y
8I:;SUMXY=SUMXY+X*Y
9I:;SUMXSQ=SUMXSQ+X*X
10I:;SUMYSQ=SUMYXQ+Y*Y
11I:;GO TO 2
12I:900;C=CORR(N,SUMX,SUMY,SUMXY,SUMXSQ,SUMYSQ)
13I:;WRITE(6,102)C
14I:102;FORMAT(' CORRELATION COEFFICIENT:',F8.3)
15I:;CALL STDEV(N,SUMX,SUMXSQ,C)
16I:;WRITE(6,103)C
17I:103;FORMAT(' STANDARD DEVIATION --X- X:',F8.3)
18I:;CALL STDEV(N,SUMY,SUMYSQ,C)
19I:;WRITE(6,104)C
20I:104;FORMAT(' STANDARD DEVIATION -- Y:',F8.3)
21I:;STOP
22I:800;WRITE(6,105)
23I:105;FORMAT(' ERROR - RETYPE LINE')
24I:;GO TO 2
25I:;END
26I:@FOR,D X.CORMAIN
LINES:25
FOR S9A-04/17/72-15:30:59 (0,)
@EOF

00000  *DIAGNOSTIC*  THE VARIABLE, SUMYXQ, IS REFERENCED IN THIS PROGRAM
      , BUT IS NOWHERE ASSIGNED A VALUE.

       END OF COMPILATION:          1 DIAGNOSTICS.
@ED,U X.CORMAIN
ED 12.02-04/17-15:32-(0,1)
EDIT
0:C /SUMYXQ/SUMYSQ/A
      SUMYSQ=SUMYSQ+Y*Y
SCAN:25
EOF:25
0:@EOF
LINES:25
```

Figure 3-12. FORTRAN Example (Page 1 of 4)

```
@FOR,D X.CORMAIN
FOR S9A-04/17/72-15:33:03 (1,)
@EOF

      END OF COMPILATION:          NO  DIAGNOSTICS.
@ED,I X.CORR
ED 12.02-04/17-15:33-(,0)
INPUT
1I:
EDIT
0:TYPE FOR
0:SET 7
0:
INPUT
1I:;FUNCTION CORR (N,SX,SY,SXY,SX2,SY2)
2I:;TX=N*SX2-SX*SX
3I:;TY=N*SY2-SY*SY
4I:;TXY=N*SXY-SX*SY
5I:;CORR=TXY/SQRT(TX*TY)
6I:;RETURN
7I:;END
8I:@FOR,D S-X.CORR
LINES:7
FOR S9A-04/17/72-15:35:40 (0,)
@EOF

      END OF COMPILATION:          NO  DIAGNOSTICS.
@ED,I X.STDEV
ED 12.02-04/17-15:36-(,0)
INPUT
1I:
EDIT
0:TYPE FOR
0:SET 7
0:
INPUT
1I:;SUBROUTINE STDEV (N,SX,SX2,STD)
2I:;STD = SQRT(SX2/N-(SX/N)**2)
3I:;RETURN
4I:;END
5I:@FOR,D X.STDEV
LINES:4
FOR S9A-04/17/72-15:37:31 (0,)
@EOF

      END OF COMPILATION:          NO  DIAGNOSTICS.
```

Figure 3-12. FORTRAN Example (Page 2 of 4)

```
@MAP,IN ,X.STATIS
MAP 0023-04/17-15:38

IN X.CORMAIN,X.CORR,X.STDEV
@EOF


@XQT X.STATIS
 ENTER DATA:
1.,1.
2.,2.
3.,3.
4.,4.
5.,5.
@EOF
 CORRELATION COEFFICIENT:    1.000
 STANDARD DEVIATION -- X:    1.414
 STANDARD DEVIATION -- Y:    1.414
END        64 MLSEC
@XQT X.STATIS
 ENTER DATA:
1.,5.
2.,4.
3.,3.
4.,2.
5.,1.
@EOF
 CORRELATION COEFFICIENT:   -1.000
 STANDARD DEVIATION -- X:    1.414
 STANDARD DEVIATION -- Y:    1.414
END        60 MLSEC
@XQT X.STATIS
 ENTER DATA:
5.,1.
10.,6.
5.,2.
11.,8.
12.,5.
4.,1.
3.,4.
2.,6.
7.,5.
1.,2.
@EOF
 CORRELATION COEFFICIENT:     .575
 STANDARD DEVIATION -- X:    3.661
 STANDARD DEVIATION -- Y:    2.280
END        80 MLSEC
```

Figure 3-12. FORTRAN Example (Page 3 of 4)

```
@PACK X.
FURPUR 023A-04/17-15:44
END PACK
@PRT,T X.

SPALDING-R*X
FOR  MAIN(2)
ABS  MAIN
ALG  ALGTEST(4)
REL  ALGTEST
ABS  ALGTEST
COB  COBTEST(5)
REL  INFILE
REL  OUTFILE
REL  C$WORKCOBTES
REL  C$0101COBTES
REL  C$0102COBTES
REL  COBTEST
ABS  COBTEST
FOR  CORMAIN(2)
REL  CORMAIN
FOR  CORR(1)
REL  CORR
FOR  STDEV(1)
REL  STDEV
ABS  STATIS
END PRT
```

All produced by COBOL

Figure 3-12. FORTRAN Example (Page 4 of 4)

4. THE ED PROCESSOR See hardware/software summary

The ED processor is a text editor which allows the user to conversationally edit a symbolic file or element. It allows insertion, deletion, and replacement of text.

4.1. The ED Processor Call Statement

The ED processor is called by the control statement:

@ED,options name1, name2

All parameters are optional except 'name1'. 'name1' and 'name2' may specify either data files or program file elements (filenames must be followed by a period).

The allowable options are:

B - Batch mode when using a demand terminal: ED will not solicit input from the user.

D - Demand mode from a batch run: output listing will contain solicitation messages.

I - Initial insertion of symbolic input from the runstream which causes ED to enter input mode. The images following the @ED statement are inserted into 'name1'; 'name2' is ignored. The I option takes precedence over the R or U option.

L - Print all lines following the @ED statement. The lines printed are indented and preceded by three asterisks.

N - Suppress printing of changed, located, or moved lines. This option serves the same purpose as the ON BRIEF command.

R - Input is taken from 'name1', no output text is produced (read-only mode). 'name2' is ignored.

U - Update the symbolic element 'name1' by applying corrections and create a new symbolic element cycle. The output element name is the same as 'name1' but with a cycle number one greater. If 'name1' names a data file in System Data Format, the original images will be replaced with the updated ones. 'name2' is ignored.

X - Kill the run (batch only) if any errors are encountered.

If none of the options I, R, or U is specified, the R option is assumed for elements and the U option is assumed for data files.

4.2. ED Processor Usage

ED operates in two modes: input and edit. In input mode, all lines entered are directly inserted into the text. In edit mode, various commands may be used to modify existing text. Changing between modes is accomplished by entering a blank line (except when ON EOF is in effect). Most editing commands implicitly reference a particular part of the text. This is accomplished by an internal cursor maintained by ED. This cursor may be positioned directly by some commands (e.g., <number>, + <number>) and indirectly by others (e.g., LOCATE, FIND).

ED proceeds sequentially through the text. It is therefore more efficient to perform editing operations in sequential order, starting at the beginning of the text.

If the user wishes to interrupt the processor, he may depress the BREAK key at any time. The system will respond with: INTRPT LAST LINE. The user should answer with a carriage return if he wishes to escape the current command. A few lines of backed-up printout may follow; then ED will request a new editing command.

Files with names of the form ED$xx (where 'x' is any character) should be avoided since ED uses such names internally.
Mail files are ED$MØ, ED$M1, etc.

4.3. Editing Commands

Following is a list of commands that may be used when ED is in EDIT mode. All commands may be abbreviated to the first three characters. Some have shorter forms; these are indicated by surrounding the shorter form with parentheses.

1. A) <number>
   B) + <number>
   C) - <number>

These commands are used to position the editor at a desired line in the text. Form (A) directs the editor to line <number>. Form (B) directs the editor to move to the position current line plus <number>. Form (C) directs the editor to move to the position current line minus <number>. When the specified line is located, it is typed out if in VERIFY mode, and modifications may be made to it.

2. A) (L)OCATE <string>
   B) (L)OCATE <quote char> <string> <quote char>
   C) LC <string>
   D) LC <quote char> <string> <quote char>

This command is used to search the text for a given string of characters. The search begins at the line following the current line and proceeds sequentially through the text until a find is made or the end of file is encountered. Form (A) ignores multiple blanks in the images. Form (B) requires that the text image be exactly the same as the string within the two quote characters. Forms (C) and (D) behave in the same manner respectively as (A) and (B) except that all occurrences of the string in the remaining text are located. Just before each line containing an occurrence is typed out, the line number is typed out.

Example: LOCATE 'AP E'

Will locate AP E but not AP E in the text.

3. LCHAR <char>

This command sets the quote character for the locate command. The default character is quote ('). A non-inputtable character will be assumed if <char> is a blank.

4. (C)HANGE <dliml> <stringl> <dliml> <string2> <dliml> <number> <GI>

This command searches a specified number of text lines for <stringl> (as with LOCATE command except that CHANGE starts with the current line and LOCATE starts with the following line). When and if the desired string, <stringl>, is found, <string2> is substituted for it. The number of lines to be scanned is indicated by <number>. The global indicator, <GI>, tells whether to change all occurrences of <stringl> in the range of lines or just the first occurrence in each line. A 'G' means all occurrences and any other character (or no character) means just the first. <dliml> may be any character which does not occur in <stringl> or <string2> except for a blank. Instead of using <number> and <GI>, a user may change all subsequent occurrences in the file by using the word 'ALL' (abbr. A) where <number> is usually specified.

5. CLIMIT <column number>

This command allows the user to set a limit on the number of columns which will be searched during performance of the change command. This is useful for protecting areas of data in a file. The default value is 132.

6. (IN)LINE <number> <termination substitute>

This command allows inline editing of a given line. If <number> is blank, the current line is assumed to be the one to be edited. Otherwise, the editor proceeds to line <number>. The line will be printed out. The user can then enter editing information directly below the line to modify it. Following are the editing characters to be used.

I - The string following this command is inserted following the character immediately above the I. The string is delimited on the right by the termination character '!' .

Example:

```
*INLINE
++APPLE=GREENG
+           IIN!
```

results in:

```
APPLE=GREENING
```

D - The characters in the line above are deleted between the D and the !

Example:

```
++APPLE=GREENING
+           D   !
```

results in:

```
APPLE=GREEN
```

R - The characters following the R will replace the characters immediately above them. A ! is required to terminate replacement.

Example:

```
++APPLE=GREEN
+     R PIE  !
```

results in:

```
APPLE PIE
```

If one wished to use another character instead of ! for termination (for example, if one wished to insert a !) then an alternate symbol may be specified as <termination substitute>. This will remain in effect for this inline only.

7. A) (P)RINT <num1> <num2>
   B) (P)RINT <num1>
   C) (P)RINT!

This command is used to print out lines of text. The first form prints lines <num1> through <num2>. The second form prints the next <num1> lines. If the command is immediately followed with a + the printing starts with the next line instead of the current one (example: PRINT + 3). Form C prints the entire file from the top. If no number or recognizable symbol follows the command, a 1 is assumed; that is, the present line will be printed out.

8. A) LNPRINT <num1> <num2>
   B) LNPRINT <num1>
   C) LNPRINT!

This command behaves like the PRINT command except that each line is preceded with its line number. Syntax is the same as the PRINT command.

9. A) (Q)UICK <num1> <num2>
   B) (Q)UICK <num1>
   C) (Q)UICK!

This command prints lines with all nonsignificant blanks omitted. This provides a fast method of examining areas of the file. <num1> and <num2> are the same as on the PRINT command. Plus (+) may also be used on form B with the same meaning.

10. A) LNQUICK <num1> <num2>
    B) LNQUICK <num1>
    C) LNQUICK!

This command behaves like the QUICK command except that each line is preceded with its line number. Syntax is the same as the QUICK command.

11. A) DITTO <num1>
    B) DITTO <num1> <num2>

This command allows duplication of other lines in the file.

The duplicated lines are inserted at the present position in the file. The first form results in the one line at <num1> being inserted in the present position. The second form results in all lines <num1> through <num2> being duplicated at the present position. Care must be exercised to be sure the most current line numbers are used. At the finish of the ditto the pointer is positioned at the next line following the lines inserted.

12. (I)NSERT <string>

This command is used to insert a line following the line presently pointed at by the editor. This new line will then be the point at which the editor is positioned. The string to be inserted starts after the first blank following INSERT. If insertion of a line longer than the space remaining on the teletype is desired, the user may type:

```
INSERT+
```

then the next teletype line will be interpreted as the <string>. If the command with no image is entered when not in EOF mode (see 'ON' command) the

editor will switch to input mode. In EOF mode this simply results in the insertion of a blank line.

13. (R)ETYPE <string>

This command is used to completely replace the current line with the string following the first blank after the command. A + may be used after the command with the same meaning as with the INSERT command.

14. A) (D)ELETE <num1> <num2>
    B) (D)ELETE <num1>

This command is used to delete lines from the text. The first form deletes lines <num1> through <num2>. The second form deletes the next <num1> lines starting with the current one.

15. IB <string>

This command behaves exactly the same as the INSERT command except that the line is inserted before instead of after the current line.

16. INPUT

This command directs the editor to enter a special input mode. In this mode everything which is typed in is inserted in the file until an exit from the mode is taken. This is especially useful when large volumes of input are to be entered. Exiting from this mode is accomplished by typing an @EOF when in EOF mode (see ON and OFF commands) or a carriage return when not. Tabs are recognized in this mode.

17. EXIT

This is the command used to take a normal exit from the editor. All of the corrections will be applied to the designated file and a normal EXIT$ will be taken.

18. OMIT

This is the command to be used if the user does not want his corrections to be applied to the file on exit. The input file will remain as it was at the beginning of the editing session, and the output file, if any, will not be produced.

19. TAB <tab char>

This command is used to specify which character is to be used as a tabulator character. This character is recognized on the INSERT, IB, and RETYPE strings and is recognized on all input when in the input mode. The character is not transmitted to the file and behaves just as a tab on a typewriter. If no TAB command has been entered, a semicolon (;) is assumed as the tab character. If <tab char> is omitted from the command, or the tab feature is inhibited.

20. SET <tab1> <tab2> <tab3> ... <tabn>

This command is used to set the tabs for the commands which allow them as explained above. As many tabs as desired may be designated. Each SET wipes out all previous tabs, and so a SET with no tabs clears the tabs. If no SET has been performed a default of 11,21,40,73 is assumed.

21. TYPE <symbolic type>

The editor cannot tell what type of symbolic element is being produced. Therefore this command has been provided to allow the user to specify the type of element he is producing if he so desires. The values <symbolic type> may have are:

Type	Meaning
ELT	ELT processor
ASM	1108-1110 ASSEMBLER
COB	1108-1110 COBOL Compiler
FOR	FORTRAN V
ALG	ALGOL
MAP	COLLECTOR (MAP processor)
DOC	DOC processor
SEC	SECURE processor ← stay away from this one

Also BAS for BASIC

For untyped elements, a default of ELT is assumed.

22. A) SITE <num1> <num2>

B) SITE <num1>

C) SITE!

This command is used to direct output to an onsite printer (PR). The meanings of <num1> and <num2> are the same as for the PRINT except that if no numbers are given, form (C) is assumed. After this command is entered, a message as follows will be typed out:

HDG?

The line typed in here will be used to head the onsite output. Periods must not be used in this header as anything beyond the period will not be printed out due to @HDG implementation. After the output is done, the following will be typed:

MSG?

The user should enter the information here necessary to indicate where and to whom the output should be returned.

23. A) LNSITE <num1> <num2>
    B) LNSITE <num1>
    C) LNSITE!

This command behaves the same as the SITE command except that each line is preceded with its line number.

24. A) CPUNCH <num1> <num2>
    B) CPUNCH <num1>
    C) CPUNCH!

This command is used to punch parts or all of a file at an onsite card punch. The syntax has the same meaning as with the site command. After the command is entered, a message as follows will be typed out:

MSG?

the line typed in here will be sent to the console onsite before the cards are punched.

25. A) OPR <string>
    B) OPR* <string>

This command is used to send a message to the onsite console. Form (A) sends the message <string>. Form (B) does the same, but also solicits an answer. The strings may not be more than 50 characters or they will be truncated.

26. A) PUNCH <num1> <num2>
    B) PUNCH <num1>
    C) PUNCH!

This command is used to punch paper tape for FORM II paper tape input at a terminal which has punch and read hardware. The syntax for this command is the same as that for the print command. When the command is entered, the following response will be given:

DEPRESS PUNCH ON

The processor will then pause to allow the user to push the punch on button on the paper tape punch hardware. It is suggested that the user previously punch several rubouts on the tape to make a leader for later reading the tape. This is accomplished by switching the paper tape punch on and holding the repeat key (REPT) and the rub out key down simultaneously. This may be safely done even while executing a run at the terminal since rubouts are ignored by the EXEC 8 software. The user should backspace the paper tape punch over one (1) rubout so that the initial carriage return will not be considered as a blank line. After pausing the designated lines will be typed out which will cause the paper tape to be punched at the same time. When the typing is finished, the editor will again pause to allow the user to switch the punch off. Rubouts should also be used at the end of the tape. The tape so produced can be used as normal FORM II input.

27. A) ON <special mode>,...,<special mode>
    B) OFF <special mode>,...,<special mode>

This command is used to define some special modes within the editor. ON turns the mode on, and OFF turns it off. The special modes are:

QUICK	-	compress extra blanks out of all output to device.
BRIEF	-	do not echo corrected images for CHANGE and DITTO.
NUMBER	-	precede each line printed out with its line number.
PCNTRL	-	recognize and print print control images.
DSPLIT	-	delete lines transferred by SPLIT command.
XBRIEF	-	do not echo lines transferred by SPLIT or ADD commands.
SEQ	-	print sequence numbers when soliciting input.
LOOK	-	look for mail after each command is executed.
EOF	-	special mode where blank lines may be entered. INP command enters input mode and @EOF exits from input mode to edit mode. While in input mode blank lines may be entered. Also the insert command with no image following will enter a blank line.
MEMORY	-	remember modes on successive executions.

All of the modes may be abbreviated to one letter.

28. RP <number>

This command is used to set a repeat counter for the INSERT command. Any insertion will be repeated <number> times.

29. A) ADD <EXEC 8 element or file designation>
    B) ADD <EXEC 8 element or file designation> <num1> <num2>

This command is used to add all or portions of a file to the current file. Form (A) adds the whole file, and Form (B) adds lines <num1> through <num2> to the current file. The lines to be added are inserted at the end of the file unless A + immediately follows the command in which case the lines are inserted following the current position within the edit file. The <EXEC 8 element or file designation> is the element, version, filename, qualifier or whatever is needed to identify the element or file according to standard EXEC 8 dropout rules.

30. A) SPLIT <EXEC 8 element or file designation>
    B) SPLIT <EXEC 8 element or file designation> <num1> <num2>

This command is used to build new elements or files from portions of a current file. Form (A) causes all the lines preceding the line currently pointed at to be reproduced as the designated file. Form (B) causes lines <num1> through <num2> to be reproduced. An ! immediately after the SPLIT command causes the whole file to be copied.

31. A) (F)IND <mask>
    B) FC <mask>

FIND searches for an image which corresponds exactly column for column starting at column 1 with the <mask>. Transparent characters may be in the mask which will test successfully with any character in the column. The normal transparent character is a blank, but alternate ones may be designated with the TCHAR command. The search begins with the line following the current one and proceeds until a match or end-of-file is detected.

Example:

Let us suppose we wanted to find a label whose form is

AB?CD

where ? can be a number from 1 to 9. The following command will find the first occurrence of such a label:

FIND AB CD

The FC command behaves in the same way as the FIND command except that all occurrences are flagged for the rest of the file.

32. TCHAR <character>

This command is used to set an alternate transparent character for the FIND command. Only one character at a time can be designated as transparent.

33. APPEND

This command positions the editor at the last line in the file and changes to input mode. This is useful when building up a file.

34. SEQ, <ID> <increment> <starting value>

This command is used to apply sequencing to columns 73-80 of the file. <ID> is a two letter identification applied to columns 73-74. <starting value> is the value at which to start numbering the remaining columns, and <increment> is the increment to use for each card.

35. TIME

This command prints out the date, time, and cycle information.

36. CPT

This command prints out the CPU time used so far in the present run. The form of the message is (minutes)M and (seconds)S.

37. SCALE <number>

This command causes a line to be printed out which can be used for column sensitive operations. The form of the line is:

12345678901234567890123456789O...

Starting in column <number>. If <number> is omitted, one (1) will be used.

38. CCHAR <char>

This command sets the continuation character. When an input line to the editor has this character in it, the editor assumes that the next line of input is a continuation of this current line. This next line will be solicited in the normal manner except that A + will precede the solicitation. The character is initially set to a character which cannot be typed in. The

character can be reset to this non-enterable character by using this command with no <char> .

39. MSCHAR <char>

This command sets a character which will be translated to a masterspace when it is input in column one; the default character is an exclamation point (!). A non-inputtable character will be assumed if <char> is a blank.

40. CSF <EXEC 8 control statement>

This command is used to submit a control statement via CSF$. Only statements valid for CSF$ may be submitted. The control statement must start in column 5.

41. EXCH <char> <octal number>

This command is used to allow input of characters not represented in the keyboard typeset. <char> is the character which is to be used to stand for the number whose internal representation is <octal number>. When <char> occurs anyplace in an input line it will be replaced by this character. An EXCH with no parameters disables this feature.

42. MAIL <userid>

This allows the user to send messages to another user. The <userid> is the nongenerated runid of the person to whom the message is directed. The editor will then solicit 10 lines of input with:

MAIL**

If the desired message is to be less than 10 lines the mode can be ended by entering an @EOF. After the message is received by the designated person, it will be deleted.

43. PCN

This command is used to enter a print control image into the file being edited. When the command is entered, the editor will solicit the image with:

CONTROL IMAGE-

This image can only be read when in a special mode set by the ON command.

44. LAST

This command causes the editor to move to the last line in the text and stay in edit mode. The last line may not be altered at this time.

45. MAXLINE <number>

This sets the maximum length to which a line may increase. If it is exceeded, the line will be truncated. The default is 80.

46. PLIMIT <column number>

This command is used to set a limit on the number of columns which will be printed out by the PRINT Command.

47. UP

This command is used to cause an element or file to be saved as if the U option was specified on the control card. This is used if the entry to the editor was made with an R option.

48. A) MOVE <num1>
B) MOVE <num1> <num2>

This command performs the same operation as the DITTO command except that the original lines are deleted after the duplication has taken place. The syntax is the same as for the DITTO command. Care must be exercised to be sure the most current line numbers are used.

49. STATUS <special mode>,...,<special mode>

This command is used to request the status of special modes set by the ON and OFF commands. If no special modes are specified, the status of all will be listed.

5. SYSTEM FAILURES

A system failure occurs when the executive determines that, due to a hardware or software error, processing cannot continue. The system is also said to have failed if the operator determines that the executive is not correctly operating. When the system fails, it is often said to have "crashed", "hung", or "died", or to be "down". Failures occur (with varying frequency) on all large-scale computer systems.

A minimum requirement for recovery from a system failure is that main storage be reinitialized. This means that all active runs are discarded, temporary files are released, program execution is aborted, and all response to demand terminals ceases. This minimum recovery sequence is normally performed without operator intervention, and takes a couple of minutes. If complications exist, it may take longer for the operator to recover the system via a process known as "rebooting." MESSY!!!!

When system recovery is complete, the message ENTER SITEID is sent to all demand terminals currently dialed in. The user must reinitiate a run, as described in sections 1.2 and 1.3. Normally, all catalogued files are recovered, so if the user has been using catalogued files as described in chapter 3, most of his work can be recovered. Of course, all data kept in main (core) storage and in temporary files is lost.

In the worst case, recovery of catalogued files is not possible. The computer center will reload all files from its latest set of backup tapes, and all user changes since that set was created will be lost. In this event, a message giving the time of backup creation and of file reloading will be displayed after all @RUN statements.

As was mentioned in section 3.8.2, files that were assigned during a system failure are marked "disabled" and the message FACILITY WARNING 000000000200 is printed on subsequent assign requests. This is primarily an aid for batch users, and generally may be disregarded by demand users. An @ENABLE filename command will remove the disabled flag from 'filename', and hence the warning message will no longer appear.

6. DEMAND TERMINAL OPERATION

Basic operational procedures were described in Chapter 1. This chapter presents details on all operating features.

6.1. Demand Terminal/System Interface Messages

The following table lists the messages and their meanings used as aids in communicating between the system and the user.

Message	Interpretation
NO RUN ACTIVE	This message is sent to the terminal whenever an image is received from it and no run has been initiated. A @RUN control statement must then be submitted to properly initiate the demand mode.
TIMEOUT TIMEOUT WARNING after 5 minutes and *TERMINAL INACTIVE* after 10.	No activity has occurred on the line for a predefined interval. If another time interval elapses without activity, the terminal is terminated and the message TIME-OUT TERMINATION is displayed at the terminal. @TAA*LIB.WAIT or @TAA*LIB.WAIT,S will prevent this.
READY Not given anymore.	Informs user that the symbiont is conditioned to receive input. This message is only transmitted if a ***WAIT*** had previously been sent to the terminal. The READY message is not sent to the terminal if output from the run is available. In either case, the wait condition is terminated.
WAIT *WAIT LAST INPUT IGNORED*	This message is sent to the terminal when: (1) An attempt is made to input from the terminal before the @RUN control statement has been completely processed (no input is accepted until the @RUN control statement is processed). The ***WAIT*** message is displayed following each character the user attempts to input. (2) An attempt is made to input from the terminal before the @FIN control statement has been completely processed (same conditions as for (1)). (3) The executive is executing programs of higher priority. (4) Both 28-word input buffers are full. The user is notified that additional input can be accepted by the READY message (no output is available) or the symbiont output to the terminal. A line image can be considered as accepted if the CR input character results in a LF/CR sequence and no ***WAIT*** is displayed.

6.2. Executive Language Interface

The demand user communicates with the system through the standard executive control statements. There are a few exceptions to the interpretation and use of some control statements when operating in the demand mode. These exceptions are:

@BRKPT Control Statement (see GTPRM, sec. 4.4.7., and sec. 8.1 of this manual) - The @BRKPT control statement is used in the same manner as it is used for a batch run. @BRKPT PRINT$/filename puts the normal print file into an SDF-formatted file.

@HDG Control Statement (see GTPRM sec. 4.4.4.) - This control statement is ignored when submitted through a demand terminal except when the output is directed elsewhere by a @BRKPT control statement.

Very useful.

@START Control Statement (see GTPRM sec. 4.4.6.) - Any run scheduled by a @START control statement submitted through a demand terminal is scheduled as a batch run. All output generated by the run is queued to the output device associated with the primary onsite card reader.

@SYM Control Statement (see GTPRM sec. 4.4.8.) - This control statement can be used to direct output to an onsite device or remote batch terminal, but not to a demand terminal.

6.3. General Operation of the Demand Symbionts

Each demand terminal supported by the operating system is controlled by a specific demand symbiont. Currently, Georgia Tech supports only the Teletypewriter Demand Symbiont. The DCT 500 (teletypewriter mode) is described as an extension of the teletypewriter symbiont.

6.3.1. Teletypewriter Demand Symbiont

The teletypewriter demand symbiont provides support for Models 33 and 35 (KSR/ASR), and the DCT 500 operating in the teletypewriter mode.

6.3.1.1. Operational Considerations

The following are features of the Teletypewriter Demand Symbiont:

(1) The symbiont accepts two forms of paper tape input (see 6.3.1.2.).

(2) Several characters are recognized by the symbiont as control sequences (see 6.3.1.3.).

(3) If a timeout occurs when a user program has a registered contingency activity, the contingency is activated and the activity is passed as an error type 2_8 and contingency type 10_8. The site's timeout process is again initialized. If no contingency is registered, the site is terminated.

(4) If a ***PARITY ERROR*** message is displayed at the terminal, the symbiont has detected a parity error on at least one character and the entire input image is discarded. Give up. Call the operator.

(5) The @TABSET control statement is available to teletypewriter users as an aid in formatting input data at the teletypewriter terminal (see 6.3.1.5.).

(6) A special routine for communications between the central site operators console and the teletypewriter terminal (see 6.3.1.6.).

6.3.1.2. Paper Tape Input

Two forms of paper tape input are permitted; they are:

Form I - Interactive Mode

Form II - Continuous Mode

6.3.1.2.1. Form I Paper Tape Input

Not used any more,

Images on paper tape consist of a string of up to 80 characters followed by the character sequence:

LF X-OFF CR DEL

where:

LF is line feed

CR is carriage return

DEL is delete (or rub out)

The DEL may not be required depending upon the teletypewriter model (experimentation may be required).

In the tape mode, all images must be in this format, or the results are unpredictable.

The paper tape mode is initiated by inserting a tape in the reader, sending the character X-ON (control Q) and on the ASR-35 models, switching from keyboard to tape mode. This causes the symbiont to send an X-ON back to the teletypewriter which then reads one image. After the end-of-image

sequence is received, any available output is sent. When the symbiont is ready to accept another image, an X-ON is sent to the teletypewriter. At no time should the teletypewriter operator manually initiate paper tape motion except by the X-ON key.

The paper tape mode is terminated by a series of two X-OFF characters in a row followed by a DEL and this causes the message 'END OF TAPE' to be displayed. These may be on the tape or entered manually.

Several of the special characters are treated differently for form I paper tape input.

BREAK - Terminates paper tape mode (no more X-ON characters are sent to the teletypewriter). The normal rules for manual input after the BREAK key input apply. Paper tape mode may be reinitiated by pressing the X-ON key. The BREAK key should not be used while the tape is in motion.

? - Causes the image in which it occurs to be ignored; however, the image must still end with the LF X-OFF CR DEL sequence.

← - Causes a one-character backspace.

LF - Needed in the end-of-image sequence to produce a readable copy on the teletypewriter printer. The LF is never considered part of the image text and is treated like a DEL.

If a tape is improperly formatted, or if characters are typed in manually while in the tape mode and the symbiont is not ready for more input, the tape mode is terminated and the message ***WAIT*** is sent to the teletypewriter. The tape mode may be reinitiated with X-ON.

If, for any reason, no input or output occurs for more than five minutes, the tape mode is terminated and the message TIMEOUT is displayed. If no further action occurs within another five minutes, the site is terminated. A tape which ends without the end-of-tape sequence can cause this since the symbiont will have sent a request for input (X-ON) and cannot do output until the request is satisfied. This problem may be cured by inserting three X-OFF's manually.

The control statements @RUN and @FIN should never appear on a paper tape, except that a @RUN control statement may occur while in the @DATA mode (see GTPRM sec. 6.5.).

The model 33 teletypewriter must have the option which allows the teletypewriter to initiate tape motion by sending an X-ON to the teletypewriter. This feature also includes the ability to have the tape stop when an X-OFF is read.

6.3.1.2.2. Form II Paper Tape Input All of this is no longer any good, See below,

Form II provides for continuous paper tape input with no interaction until the end-of-tape signal is received. The images are buffered on mass storage. At end-of-tape, the buffered input is internally added to the input stream.

The only requirement as to the image format is that a CR character mark the end-of-image. A LF, however, is useful for monitoring the tape as it is read. Without a LF, overprinting of each image will result. No X-ON or X-OFF characters are needed or desirable on the tape.

The following procedure is used for form II paper tape:

(a) The user must have an active run. If an attempt is made to read form II without a run active, the following message is printed at the terminal:

NO RUN ACTIVE

(b) To start the tape input, the terminal operator must press keys CNTL and TAPE. The message

TAPE START

is printed at the terminal. If the tape is in the reader and the reader MODE switch is set to AUTO, the tape is read automatically. If the tape is not in the reader, the operator must place it there and set the reader MODE switch to MANUAL READ.

(c) When the tape read is completed, the terminal operator then must press keys CNTL and TAPE with LINE. (i.e., CNTL and T). The message

END OF TAPE

is printed at the terminal. Tape input images are now added to the input stream.

To start paper tape, type @@PTI. To end paper tape input, type @@END.

The @RUN and @FIN control statements are disregarded if they are on the input tape. It is suggested that neither should be used, except when using @DATA or @ELT,D control statements (see GTPRM, 6.5 and 6.4, respectively).

The EOT character is recognized in the form II mode. The terminal run is terminated as in manual mode.

The rub-out character may be used to rub out errors during preparation of the tape. The rub-out is ignored by the handler as are nulls.

The question mark and left arrow are not recognizable as delete characters.

6.3.1.3. Special Control Sequences

The following table lists the teletypewriter control characters and their functions. These control characters are used to control image formats, image input, and so forth.

Keyboard Key	Function	Description
?	Delete	When received from the terminal, the current image is discarded. The symbiont responds to delete function with a CR/LF sequence.
RETURN	End of Image	Used to indicate the end of the input image. Maximum input image length is 80 characters. The symbiont responds with a LF.
EOT	End of Transmission	Terminates and disconnects the teletypewriter terminal.
←	Erase	One preceding character is deleted each time the key is pressed. Characters are deleted right to left.
ESC	Escape	When the key is pressed, the next character and only the next character is inputted in the escape mode. This control character allows the user to input the ? character as data.
BREAK or RTS	Interrupt	Causes the symbiont to suspend its current operation and accept an input image immediately. The message INTERRUPT LAST LINE is sent to the terminal. When the user keys in the CR following his input image, the line of output that was interrupted is immediately resumed. (See 6.3.1.4.)

6.3.1.4. BREAK Key obsolete. See below.

The BREAK key character is represented on the keyboard as BREAK or RTS. The BREAK key may be used at any time other than within an input image; that is, once an input image has been started, a CR or a ? (line delete) must be sent before the BREAK key can be used. If used within an input image, the BREAK key is inserted in the image as an unknown character.

Upon receiving the BREAK key, the symbiont suspends its current operation and sends the following message to the terminal:

INTERRUPT LAST LINE

Teletypewriter models 33 and 35 require that the BRK/RLS key be pressed before any character can be sent from the terminal after a BREAK key.

The symbiont is now ready to receive one of three possible commands.

(1) Terminate user execution. This is accomplished by the character X followed by a CR. If the X is received while in the @ADD mode, all @ADD files and backed-up input are discarded. A few lines of backed-up output may be printed. The next input is then expected from the terminal. Use @@X TIO.

(2) Contingency interrupt. The contingency routine specified for the user's run is given control with the error code of 08 in the contingency status word when any character other than an XCR (for example, just a CR) is received by the symbiont. If a contingency routine has not been specified, the execution is terminated as if the BREAK key was followed by an XCR. Use @@X CO

(3) Remove a run from a facilities-held status. From a terminal, the BREAK key followed by the keyin of an X may be used to take a demand run out of the hold status due to its facility requirements. All backed-up input including the facilities request is discarded. @@X TIO

If the terminal operator should decide not to enter a command or if he decides he has made an error after pressing the BREAK key, he may use the question mark(?) character to signal the symbiont to disregard the BREAK key interrupt. Use @@CONT, It works on CRT's or TTY's.

6.3.1.5. Tab Control Statement (@TABSET) No longer in use.

The demand terminal user can define horizontal tab columns for the input by use of the @TABSET control statement. The only optional parameter is the label parameter. The format of the @TABSET control statement is

```
@TABSET x1,x2,...,x18
```

where each 'xi' is the numeric specification of the tab column ranging from 1 through 80, specified in ascending order. A maximum of 18 tab positions may be specified. The following is an example of a @TABSET control statement:

```
@TABSET 3,20,30,57,60
```

Once the @TABSET control statement has been introduced from the terminal, it is in effect until either another @TABSET control statement or a @FIN control statement is received by the symbiont.

If the @TABSET control statement is accepted (no error message), the tab character (press the CNTL and I keys on the keyboard) spaces the next character of input to the next position specified by the @TABSET control statement.

Should the symbiont encounter the tab character when a @TABSET definition has not been specified or when the last defined position has been exceeded, the character is placed in the input image as a blank.

If the @TABSET control statement is in error, the message

```
TAB STATEMENT ERROR
```

is displayed on the terminal and any previous tab definition is ineffective.

6.3.1.6. Central Site to Remote Site Operator Communication

Two unsolicited keyins are available which enable the central site operator to initiate remote site communications.

The teletypewriter broadcast keyin can be used to display the specified text at all active teletypewriter terminals.

The teletypewriter message keyin can be used to display the message at the active terminal specified in the keyin.

Except when the operator specifies otherwise, the broadcast or teletypewriter message is not displayed until after execution of the current task or when the system encounters a @RUN control statement.

The user may send a message to the onsite operator via the @@MSG statement. The format is

@@MSG text

The 'text' will be prefixed by the siteid and displayed immediately, regardless of backed-up input, program execution, etc. The @@MSG is completely transparent in that it never causes an end-of-file condition.

6.3.1.7. DCT 500 in Teletypewriter Mode

A DCT 500 operating in teletypewriter mode is very similar to teletypewriter operation. There are, however, some minor considerations.

(1) The DCT 500 must be strapped to appear to the system as if it were a teletypewriter. Specifically, the DCT 500 hardware must have the following:

   (a) The RID, SID, and STX feature must be inhibited.

   (b) The parity select feature must be set to ignore parity on data received from the system.

   (c) The DCT 500 must be in the master mode.

   (d) The DCT 500 full/half-duplex option must be set to the half-duplex mode.

   Once the terminal has established a line connection with the central site, the terminal operator must depress the PROCEED key to establish clear-to-send at the DCT 500. The CLEAR TO SEND indicator lights if the data set is in data mode when the PROCEED key is pressed. Once this sequence is performed, the terminal operator can send a siteid to the system.

(2) The DCT 500 has the capability of generating the full ASCII character set; however, the teletypewriter symbiont does not handle the full ASCII set. Lower case characters are translated as upper case characters, that is, a lower case a and an upper case A will produce a Fieldata A after translation by the teletypewriter symbiont. It should be noted that the idle line logic does not recognize lower case ASCII; therefore, the terminal operator must key-in the alphabetic characters of the siteid

in upper case. There is no upper case for the ASCII numerics. The second character of siteid must be a D to signify that the terminal is a DCT 500. The teletypewriter symbiont allows the DCT 500 to receive up to 132 characters of output per line.

(3) Whenever the terminal operator desires to utilize the break (interrupt last line) feature, press the INTRPT key. This key is analogous to the BREAK key on the teletypewriter.

(4) When submitting form II paper tape from the DCT 500:

(a) The teletypewriter CNTL-TAPE character is a CTL-R on the DCT 500 and the CNTL-TAPE-LINE character is a CTL-T.

(b) The teletypewriter erase character is the shift with underline on the DCT 500.

7. ADVANCED DEMAND TERMINAL TECHNIQUES

This chapter deals with executive control statements that help the user efficiently use a demand terminal.

7.1. The @ADD Statement

Extremely useful.

If a sequence of images are to be repeatedly entered, it is possible to enter them once and subsequently have one @ADD statement represent all such images in the runstream. The images may be source language, data, control statements, or any combination.

The ED processor (Chapter 4) is recommended for placing the images in a file or element. The MSCHAR feature of ED can be used to enter control statements.

After the file or element has been created, the same effect as typing in the images may be achieved by entering:

```
@ADD filename.
or @ADD filename.eltname
```

as appropriate. The @ADD statement itself does not cause an end of file condition for an executing program.

Some warnings: If the @ADD file or element is being updated via ED, be sure to terminate ED before doing the @ADD. If this is not done, the file or element without the latest corrections will be added. Also, @ADD of an element containing an @ASG of the file it is in accomplishes nothing, because the exec returns the file to its original assign status after the @ADD has been fully processed.

Additional details are in GTPRM, section 4.4.5.

7.2. The @START Statement

Also good.

The @START statement causes the exec to treat the images in a file or element as if they had been read in from an onsite card reader. @START is recommended when lengthy, non-conversational processing is to be done.

Again, the ED processor is recommended for building the file or element.

```
@ED,I X.COMPILE
ED 12.02-04/17-15:49-(,0)
INPUT
1I:
EDIT
0:MSCHAR /.
0:
INPUT
1I:/ACOB,BE X.COBTEST
2I:A
3I:/PREP X.
4I:/MAP,IN ,X.COBTEST
5I:IN X.COBTEST
6I:LIB X.,SYSTEM$*EXTRA.
7I:/XQT ?
/PACK X.
8I:/XQT X.COBTEST
9I:
EDIT
8:LNQUICK!
1:@ACOB,BE X.COBTEST
2:A
3:@PREP X.
4:@MAP,IN ,X.COBTEST
5:IN X.COBTEST
6:LIB X.,SYSTEM$*EXTRA.
7:@PACK X.
8:@XQT X.COBTEST
SCAN:8
EOF:8
0:@EOF
LINES:8
@XQT?
@ADD X.COMPILE
COB00A5-04/17-15:53-(7,)

THE A AND B MARGINS WILL BE COL. 1 AND 2,(NO SEQ. NUMBERS)
CONTINUATION MARK MUST BE IN COL. 1, TEXT OF CONTIN IN COL 2
ABOVE FORMAT APPLIES TO..A-NONE,B-TTY,C-TTY+PF INPUTS
ERROR   0046            NUMERIC MOVE MAY RESULT IN LEFT TRUNCATION
ERROR   0045            NUMERIC MOVE MAY RESULT IN LEFT TRUNCATION

END  COB  ERRORS:       0

COMPILE TIME IS  0001.16 SECONDS
FURPUR 023A-04/17-15:54
END PREP
MAP 0023-04/17-15:54
```

@ELT,OI
is faster, since you don't need a MSCHAR. Just type in your commands.

@DATA,OI
can be used for files.

Figure 7-1. Example of @ADD

```
FURPUR 023A-04/17-15:54
END PACK
ENTER NUMBERS TO BE AVERAGED
9999V999 IS THE PICTURE
```

The file or element must begin with a @RUN statement. No @PWRD statement may be used. The end of file or element is treated as an implied @FIN statement.

The 'runid', 'reference-number', and 'user-name' for the started run are taken from the starting run. This ensures proper disposal of the onsite printout and prevents unauthorized use of a reference number.

There are three rules that must be followed to ensure a proper start:

(1) The file being started, or the program file containing the element being started, must be @FREE'd before the @START. Otherwise, the starting run may have the file exclusively assigned and prevent the exec from reading the file.

(2) Any files referenced by the started run must be @FREE'd before the @START. Otherwise, the run may abort or go into a facilities wait.

(3) The started runstream may not reference the file it is in. Since the exec has the file assigned to read the runstream, the run can never gain exclusive use of the file. The run will abort or go into an unresolvable facilities wait.

The simplest forms of the @START statement are:

```
@START filename.
```

and

```
@START filename.eltname
```

Additional fields allow replacement of certain fields on the @RUN statement in the file or element. See GTPRM section 4.4.6 for further details.

7.3. The @BRKPT Statement

The @BRKPT statement allows the user to send lengthy output to the central site high-speed printers. Convenient demand usage of @BRKPT may be achieved by use of the BRK processor, described in section 8.1.

7.4. Considerations for Public Files

The following sequence should be used to set up a file for use by many users:

```
@ASG,CPR  qual*file//wkey,F2
@COPY,P   master.,qual*file.
@FREE     qual*file
```

The C option on the @ASG means the file is to be catalogued. The P option means that any user may assign the file (public). The R option means the file is to be read-only; this will allow many users to have the file assigned "exclusively" (as is required by many system components) at the same time.

The qualifier, 'qual', and 'file' should be selected to be descriptive and easily spelled. 'qual' should be explicitly specified, since if it is omitted the user name will be used, and most user names are hard to spell.

'wkey' is the write key. The file is read only and hence cannot be written into regardless if the key is known; however, the key must be known in order to delete the file.

The @COPY transfers the desired information into the file. It is a good idea to build up the information in a normal file (one that can be easily changed), and once it is checked out transfer it into the special public file. Omit the P option if the file 'master' is not a program file.

The @FREE performs final cataloguing action for the file, including making it read only. Note that @CAT should not be used instead of @ASG,C to create a read only file, since the file will be immediately read only and hence can never contain any data.

If changes become necessary, they should be made in the file 'master'. Then 'qual*file' should be deleted and the above sequence redone. However, if anyone is using the file, it cannot actually be deleted until everyone using it is through. In this case, the following sequence may be used:

```
@ASG,CPR  qual*file(+1)//wkey,F2
@COPY,P   master.,qual*file(+1)
@FREE     qual*file(+1)
```

This creates the next F-cycle, and anyone subsequently referencing 'qual*file' will get the latest. Beware, though, that no more than 32 F-cycles may ever be created for a 'qual*file' combination.

One other note: if the file contains subroutines (relocatable elements) and is to be referenced on a MAP processor LIB directive, don't forget to @PREP qual*file after the @COPY and before the @FREE.

8. CONVERSATIONAL SYSTEM PROCESSORS

System processors differ from language processors in that they perform system-related functions, such as file handling. The GTPRM, Chapter 6, describes several system processors (e.g., MAP, LF) that are useful in both batch and demand modes. This chapter is devoted to complete descriptions of system processors whose primary usage is in the demand mode.

The ED processor can be considered to be a conversational system processor, but it is described in a separate chapter (Chapter 4) because of its singular importance.

8.1. BRK - Demand Breakpoint Interface

The @BRKPT PRINT$ feature available for demand runs can be quite useful for handling large print files. By breakpointing the print file, the user can cause printout to be placed in a mass storage file rather than being typed on the terminal. Subsequently, the file can be printed onsite or scanned with the ED processor.

The BRK program merely generates the EXEC 8 control statements necessary to accomplish the breakpoint, thus saving the user some typing.

To cause printout to be placed in a file,

@BRK,F Just use @BRK

is typed. The response will be

OUTPUT DIVERTED TO FILE filename

where 'filename' is a mass storage file catalogued by BRK for the user.

To return print mode to the terminal,

@BRK,T use: {@BRK,N if you don't want it printed / @BRK,Y if you do.

is typed. The response will be

PRINT FILE filename NOW?

If the user's answer is Y, 'filename' will be @SYM'd and decatalogued. If the answer is N, no further action is taken and it is the user's responsibility to dispose of 'filename' as he sees fit. It can be edited, etc.

Each @BRK,F call should be paired with a @BRK,T call.

A call on @BRK,F is equivalent to entering:

```
@ASG,UGP    file,F2///3000
@BRKPT      PRINT$/file
```

A call on @BRK,T is equivalent to entering:

```
@BRKPT      PRINT$
@FREE       file
```

And, if the Y response is given,

```
@SYM        file,,PR
```

8.2. FED - Conversational File Editor

I don't know whether this works or not.

This program allows the user to examine, and if necessary, to modify the contents of any mass storage file in any format. It deals strictly with sectors and tracks, and makes no attempt to conform to any particular data format. Observations may be taken in octal or alphabetic format, while corrections may be made in any combination of octal, alphabetic, integer, or floating point.

The processor call is:

```
@FED
```

The program will respond

```
GENREL FILEDIT LEVEL X
FILENAME?
```

Answer the 'FILENAME?' question with the complete filename, including, if required, qualifier, read key, write key, etc. FED will assign the file if it is not already assigned, print the status if non-zero, and then query:

```
FUNCTION?
```

This question can be answered in a number of ways. Possible answers are listed below.

GET n	Sector 'n' (octal or decimal integer, octal denoted by a leading zero) is loaded into the sector buffer.
ALPHA m,n	Prints 'n' words beginning with word 'm' in alphabetic format. The words in a sector are numbered from 1 to 28. Any attempt to print a word numbered less than 1 or greater than 28 will result in an error message.
OCTAL m,n	same as the ALPHA command, except that it prints in an octal format.
CHANGE m,w1,w2....	Changes the contents of the sector buffer beginning with word 'm'. The words 'w1,w2,...' may be any combination of octal (leading zero), floating point (decimal point in number), integer, or alphabetic (delimited by quotes). An alphabetic item which is longer than 6 characters will occupy more than one word. The words 'w1,w2,...' are written over the existing contents of words m,m+1,... in the sector buffer. The mass storage file itself is not changed until the execution of a WRITE command.
WRITE	Writes the current contents of the sector buffer back to mass storage. This command must be executed after any change commands if the changed values are to be saved.
NEXT NEXT n	Loads the sector buffer with the contents of the 'n'th sector after the current one. If 'n' is left blank, it is assumed to be 1.
TOP	Loads the sector buffer with the contents of sector zero. This command is primarily used in conjunction with the SEARCH command described below.
SEARCH target	Searches the mass storage file for the next occurrence of the item 'target' following the current sector. The search target may be any number of words long and may be a comma-separated mixture of octal, decimal, floating point, and alphabetic items. See the CHANGE function above for a description of the format for entering the target

as octal, etc. The search is performed up to and including the track specified in the LENGTH directive. If no LENGTH directive has occurred, a length of 64 tracks is assumed.

LENGTH n — Declare the length (in tracks) of the working file, for SEARCH purposes only.

TRACK n — Loads the sector buffer with the first sector of track 'n'.

(blank line) — A blank line in response to the 'FUNCTION?' question causes FED to re-ask the question 'FILENAME?' after returning the old file to its original assign status.

END — Typing 'END' will terminate FED, and return the current file to its original assign status.

In all cases, the function names may be abbreviated to the first character.

8.3. FASG - List Files Assigned

The FASG program prints a tabular listing of all files and @USE names currently assigned to the run.

FASG is called by

```
@FASG
```

@PRT,I is faster, but not as good.

or by

```
@FASG,options
```

The following options are available:

A - Only list assigned files.
U - Only list @USE relations.

In the absence of both the A and U options, both assigned files and @USE relations will be listed.

S - List TPF$, DIAG$, and SYS$*LIB$.

Lack of the S option will inhibit listing of these files.

L - Dump file descriptors and granule tables.
M - Dump file descriptors.

The user would not normally be concerned with the L or M options.

The following information is printed about each file:

Read/write permissions:

R - read only
W - write only
N - neither read nor write
(blank) - both read and write

Status:

T - temporary
A - catalogued
C - being conditionally catalogued
U - being unconditionally catalogued
D - being conditionally deleted
K - being unconditionally deleted

8.4. CPMD - Conversational Post Mortem Dump

The Conversational PMD is a program designed to allow the demand terminal user all the conveniences of a full core dump (and then some), with none of the obvious drawbacks. For example, most core dumps are taken in octal, but most of the real information to be gleaned from the dump can be gotten only by converting the octal numbers to a different format. The CPMD program will automatically convert to any one of a number of useful formats, thus saving the tired programmer the work of conversion.

When a program terminates on the 1108 system, the final contents of its memory are written to a file whose name is 'DIAG$'. The CPMD program allows the demand user to selectively examine the contents of this file, which is equivalent to selectively examining a full core dump.

The processor call for CPMD is simply:

```
@CPMD
```

with no options or fields. The program will respond with the level of the CPMD program and the lower and upper storage limits of the program's instruction and data banks. Next, input is solicited with the typeout:

```
FUNCTION?
```

A description of the permissable functions is given below. In all cases, the function names may be abbreviated to the first 2 characters. In addition, A may be used for ALPHA; O for OCTAL; and I for PROGRAM.

OCTAL

The OCTAL command causes CPMD to print out the contents of selected cells of memory in an octal format. Four words of twelve digits each are printed on each line. The format of the command is:

OCTAL m,n

where 'm' is the starting address and 'n' is the count of consecutive words to be dumped. If the field 'n' is omitted, it will be taken as one. The field 'm' is always treated as octal, regardless of the presence or absence of a leading zero. The address 'm' may be relative or absolute. See the section entitled "Relative Dumping Mode" for a description of relative addresses.

ALPHA

The ALPHA command causes CPMD to print out the contents of selected cells of memory in an alphabetic format. Eight words of six characters each are printed per line. The format of the command is:

ALPHA m,n

where interpretation of the 'm,n' field is the same as for the OCTAL command.

FLOATING

The FLOATING command causes CPMD to print out the contents of selected memory cells in an edited floating-point (REAL) format, five numbers per line. The calling format and restrictions are exactly as described above for OCTAL.

INTEGER

The INTEGER command causes CPMD to print out the contents of selected cells of memory in a base 10 integer format. Five numbers are printed per line. The format and restrictions are as described above for OCTAL.

PROGRAM

The PROGRAM command causes CPMD to print out the contents of selected cells of memory in a reconstructed assembly language format. Operation

mnemonics and register names are printed. The format is one instruction per line. When relative addressing mode is used, all U-field addresses printed which reference the same relocatable element that contains the instruction are un-relocated and printed as 'address/location-counter' rather than 'address'.

MAP

The MAP command causes CPMD to access the diagnostic tables in the absolute element from which the program was executed and from these tables to reconstruct the storage allocation map which resulted from the program collection. Only user-generated subroutines and the main program will be listed--no information about library and system subroutines will be printed.

If a relocatable element name is specified on a MAP command, then only the allocation information for that element will be printed.

LMAP

The LMAP command is identical to the MAP command except that it prints all subroutines, both user-generated and system library-provided.

DECK

The DECK command is used to specify a deck name (i.e., relocatable element name) to be used as the base of relative addressing. (See the section on relative addressing). All relative addresses are taken as relative to the specified deck name. The call is simply:

```
DECK deckname
```

where 'deckname' is the 1 to 12 character relocatable element name.

IDENTIFY

The IDENTIFY command causes CPMD to search its diagnostic tables in an attempt to determine the deck, location counter, and relative address corresponding to the specified absolute address. The call is:

```
IDENTIFY m
```

where 'm' is an absolute address in octal. If 'm' is in the program being dumped, the CPMD will print out the corresponding deck name, location counter, and relative address; otherwise, an error message.

ADD

The ADD command causes CPMD to add a list of numbers and print the sum in both octal and decimal. It may be used for subscript calculations, link tracing, etc. If only one number is provided to be added, it will be printed out in both octal and decimal, thus serving as a converter.

DIMENSION

The dimension command is used to specify dimension information for use with the SUBSCRIPT command. The call is:

```
DIMENSION a,b,c,...
```

where 'a', etc. are decimal integers.

SUBSCRIPT

The SUBSCRIPT command computes multiple subscripts for FORTRAN arrays. Suppose, for example, that one wanted to dump VAR(4,2,1), where VAR had been dimensioned to (5,6,7). The statement

```
DIMENSION 5,6,7
```

would enter the appropriate dimension information. Then, typing

```
SUBSCRIPT 4,2,1
```

will cause CPMD to print the equivalent linear subscript. Using the ADD command to add the base address to the linear subscript will give the address to dump.

Relative Dumping Mode

If the address portion of a dump command (shown as 'm' in the above descriptions) is coded as 'address/counter' instead of just 'address', then the address will be interpreted as being a relative address, relative to the specified location counter in the deck declared in the last DECK command. For example, suppose that there exists a deck UUU which contained data in location counter 3 from addresses 013044 to 023266. Then the following two commands would print the same values:

```
OCTAL 13050,2
OCTAL 4/3,2
```

Thus, location 13050 is relative location 00004 on location counter 3 in the deck under study, UUU.

Example of CPMD Usage

This example shows the interaction of system components necessary for CPMD.

The complier (in this case, FOR) is used to print a storage map. For large programs, this may be done using BRK or @START to print the listing onsite.

The program is executed. Then CPMD is executed.

The first function, MAP, shows the addresses of all user-created relocatable elements. The remainder of the program was collected from SYS$*RLIB$.

The DECK function is used so that relative addresses (as appear in the compiler's storage map) may be used for element MAIN.

Relative location 27 (octal) under location counter 1 is displayed in "program" format. "Location counter," "block," and "control counter" are all synonymous in this context.

Relative locations 4 through 14 (octal) under location counter 0 are printed in "alpha" format. This is the FORMAT statement 101, indicated by 101F in the compiler's storage map.

Relative location 3 under location counter 0 is displayed in both "floating" and "octal" formats. This is the final value of the variable AVG.

Relative location 0 under location counter 0 is displayed in "floating" format. This is the final value of the variable SUM.

The final value of the variable N is displayed with the function INTEGER 1/0.

```
@FOR,S X.MAIN
FOR S9A-04/17/72-16:33:55 (1,)
@EOF

   MAIN PROGRAM

   STORAGE USED: CODE(1) 000050; DATA(0) 000030; BLANK COMMON(2) 000000


   EXTERNAL REFERENCES (BLOCK, NAME)

    0003   NINTR$
    0004   NWDU$
    0005   NIO2$
    0006   NRDU$
    0007   NSTOP$


   STORAGE ASSIGNMENT  (BLOCK, TYPE, RELATIVE LOCATION, NAME)

    0001   000027 10L        0000   000004 101F       0000   000012 102F
           0000   000014 103F       0001   000011 5L
    0000 R 000003 AVG        0000 I 000001 N          0000 R 000000 SUM
           0000 R 000002 VAL



00101      1*              SUM=0.
00103      2*              N=0
00104      3*              WRITE(6,101)
00106      4*      101     FORMAT(' ENTER NUMBERS TO BE AVERAGED')
00107      5*      5       READ(5,102,END=10)VAL
00112      6*      102     FORMAT(E20.5)
00113      7*              SUM=SUM+VAL
00114      8*              N=N+1
00115      9*              GO TO 5
00116     10*      10      AVG=SUM/N
00117     11*              WRITE(6,103)SUM,N,AVG
00124     12*      103     FORMAT(' SUM=',E10.3,' N=',I10,' AVG=',E10.3)
00125     13*              STOP
00126     14*              END

       END OF COMPILATION:        NO  DIAGNOSTICS.
```

Figure 8-1. Usage of CPMD (Page 1 of 2)

```
@XQT X.MAIN
 ENTER NUMBERS TO BE AVERAGED
5.0
10.0
15.0
20.0
@EOF
 SUM=    .500+02 N=           4 AVG=    .125+02
END        50 MLSEC
@CPMD
GENREL CPMD LEVEL 10.
IBANK: 001000 TO 010352
DBANK: 040000 TO 044030
STARTING ADDRESS: 010303
FUNCTION? MAP
BLANKSCOMMON 00 044001 044000
MAIN         00 044001 044030
             01 010303 010352
FUNCTION? DECK MAIN
FUNCTION? PROGRAM 27/1
000027  LA,XU     A0,0233
FUNCTION? ALPHA 4/0,8
000004  (029H    ENTER   NUMBER  S TO B  E AVER  AGED)   (E20.5  )
FUNCTION? FLOATING 3/0
000003    0.125000+02
FUNCTION? OCTAL 3/0
000003  204620000000
FUNCTION? FLOATING 0/0
000000    0.500000+02
FUNCTION? INTEGER 1/0
000001             4
FUNCTION? @EOF
END CPMD.
```

Figure 8-1. Usage of CPMD (Page 2 of 2)

Appendix A

UNIVAC 1108 CARD CODES

Char In Machine	Fieldata (Octal)	029 Mode			026 Mode			Key On Teletype	Char Printed On TTY
		Holes	Key On 029	Key On 026	Holes	Key On 029	Key On 026		
@ (1)	00	7-8	≥	None(3)	7-8	≥	None(3)	@	@
[	01	12-4-8	[	)	12-5-8	(	None(3)	[(shift K)	[
]	02	0-6-8	]	None(3)	11-5-8	)	None(3)	](shift M)	]
#	03	3-8	#	=	12-7-8	←	None(3)	#	#
∆	04	11-7-8	≤	None(3)	11-7-8	≤	None(3)	↑	↑
(Blank)	05	None	(Space)	(Space)	None	(Space)	(Space)	(Space)	(Space)
A	06	12-1	A	A	12-1	A	A	A	A
B	07	12-2	B	B	12-2	B	B	B	B
C	10	12-3	C	C	12-3	C	C	C	C
D	11	12-4	D	D	12-4	D	D	D	D
E	12	12-5	E	E	12-5	E	E	E	E
F	13	12-6	F	F	12-6	F	F	F	F
G	14	12-7	G	G	12-7	G	G	G	G
H	15	12-8	H	H	12-8	H	H	H	H
I	16	12-9	I	I	12-9	I	I	I	I
J	17	11-1	J	J	11-1	J	J	J	J
K	20	11-2	K	K	11-2	K	K	K	K
L	21	11-3	L	L	11-3	L	L	L	L
M	22	11-4	M	M	11-4	M	M	M	M
N	23	11-5	N	N	11-5	N	N	N	N
O	24	11-6	O	O	11-6	O	O	O	O
P	25	11-7	P	P	11-7	P	P	P	P
Q	26	11-8	Q	Q	11-8	Q	Q	Q	Q
R	27	11-9	R	R	11-9	R	R	R	R
S	30	0-2	S	S	0-2	S	S	S	S
T	31	0-3	T	T	0-3	T	T	T	T
U	32	0-4	U	U	0-4	U	U	U	U
V	33	0-5	V	V	0-5	V	V	V	V
W	34	0-6	W	W	0-6	W	W	W	W
X	35	0-7	X	X	0-7	X	X	X	X
Y	36	0-8	Y	Y	0-8	Y	Y	Y	Y
Z	37	0-9	Z	Z	0-9	Z	Z	Z	Z
)	40	11-5-8	)	None(3)	12-4-8	[	)	)	)
-	41	11	-(5)	-(5)	11	-(5)	-(5)	-(5)	-(5)
+	42	12-0	+	None(3)	12	&	+	+	+
<	43	12-6-8	<	None(3)	12-6-8	<	None(3)	<	<
=	44	0-5-8	=	None(3)	3-8	#	=	=	=
>	45	6-8	>	None(3)	6-8	>	None(3)	>	>
&	46	12	&	+	2-8	None(3)	None(3)	&	&
$	47	11-3-8	$	$	11-3-8	$	$	$	$
*	50	11-4-8	*	*	11-4-8	*	*	*	*
(	51	12-5-8	(	None(3)	0-4-8	%	(	(	(
%	52	0-4-8	%	(	0-5-8	=	None(3)	%	%
:	53	5-8	:	None(3)	5-8	:	None(3)	:	:
?	54	12-7-8	←	None(3)	12-0	+	None(3)	Note 6	?
!(Exclaim)	55	11-0	X(Times)	None(3)	11-0	X(Times)	None(3)	!	!
,(Comma)	56	0-3-8	,(Comma)	,	0-3-8	,	,	,	,
\	57	2-8	None(3)	None(3)	0-6-8	]	None(3)	\ (shift L)	\
0	60	0	0	0	0	0	0	0	0
1	61	1	1	1	1	1	1	1	1
2	62	2	2	2	2	2	2	2	2
3	63	3	3	3	3	3	3	3	3
4	64	4	4	4	4	4	4	4	4
5	65	5	5	5	5	5	5	5	5
6	66	6	6	6	6	6	6	6	6
7	67	7	7	7	7	7	7	7	7
8	70	8	8	8	8	8	8	8	8
9	71	9	9	9	9	9	9	9	9
'(Quote)	72	0-7-8	11	None(3)	4-8	@	-(4)	'(Quote)	'(Quote)
;	73	11-6-8	;	None(3)	11-6-8	;	None(3)	;	;
/	74	0-1	/	/	0-1	/	/	/	/
.(Period)	75	12-3-8	.	.	12-3-8	.	.	.	.
□	76	4-8	@	-(4)	0-7-8	"	None(3)	"	"
≠ or stop (2)	77	0-2-8	≠	None(3)	0-2-8		None(3)	Cntrl C	None(2)

"S" indicates Numeric Shift of letter following.

Notes are for standard IBM punches. Table is for Ga. Tech punches modified to accept the Burroughs Common Language (BCL

NOTES:

1) The @ is the control card flag if it appears in column 1 of any card.

2) The ≠ will not be printed on the high speed printer or the teletype. On these two devices the ≠ character acts as the line termination character.

3) None means that there is no such key. This hole pattern must be multipunched.

4) This is the underscore character that is under the = sign on the 026 keypunch.

5) This is the minus character and is marked SKIP on the 026 keypunch. Either upper or lower shift may be used.

6) Normally, the ? key causes deletion of the current line. To enter ? as an input character, press ESC (ESCAPE), then the ? key.