

CHAPTER 3 USING MiniMINC

This chapter both explains and demonstrates how to communicate with the MiniMINC programming system.

MiniMINC performs a wide spectrum of arithmetic and trigonometric functions. It's capable of providing immediate responses, like a calculator, or solutions that depend on complex formulas and variables.

CALCULATING WITH MiniMINC

Should you require a number's square root (SQR) or log (LOG or LOG10) or the sine of an angle (SIN), type PRINT, the calculation, and press the RETURN key. MiniMINC responds immediately.

If, for instance, you want the square root of 5378, type:

```
PRINT SQR(5378) (RET)  
73.3348
```

and the system returns the answer on the next line. This method applies for almost any calculation, including addition and subtraction.

```
PRINT 23 + 6 - 15 (RET)  
14
```

Use the DELETE key to correct typing mistakes before pressing RETURN.

You can type "PRINT" once for multiple calculations, separating the different elements with commas or semicolons. For example:

```
PRINT 2+4,5-3,24/8 (RET)
6          2          3
```

prints all the answers on the same line, reserving 14 columns for each answer (there are either 80 or 132 columns to a blank line). If you use semicolons, answers are displayed on the same line, separated by one or two spaces. For example:

```
PRINT 2+4;5-3;24/8 (RET)
6  2  3
```

There's no need for you to retype values that remain constant throughout a multiple calculation. Label each constant with its own letter from A to Z, using the form Letter = Value, and then type the letter in the calculation instead of the constant. For example:

```
A=2.1362 \ B=435 \ PRINT A+B,A-B,SQR(B) (RET)
437.136          -432.864          20.8567
```

These letter/number associations remain intact in MiniMINC's memory until you assign a different value to any of the letters, erase memory, or shut off MiniMINC. The backslashes (\) tell MiniMINC where one statement ends and another begins.

ERROR MESSAGES

MiniMINC checks the syntax and internal consistency of a command before executing it. Should you misspell a word or omit some necessary part of the command's structure (described in a later section of this chapter), MiniMINC usually responds with an error message instead of performing the command. Error messages announce mistakes or problems; MiniMINC even produces some messages that recommend what you should do. An error message occurs if you type the following:

```
PRUNT SQR(5378) (RET)
```

You misspelled PRINT. MiniMINC fails to recognize PRUNT as PRINT and responds:

```
?MINC-F-Syntax error; cannot translate the statement
```

MiniMINC issues messages that identify the error specifically (?MINC-F- Invalid PRINT USING format or syntax) or generally (?MINC-F-

Syntax error; cannot translate the statement). Sometimes MiniMINC halts system activity until you decide between two courses of action (?EDITOR-W- Abort edit session losing all edits (Y,N)?); at other times, you must proceed on your own initiative.

You are likely to make a great number of mistakes as a person new to MiniMINC. In fact, everyone is expected to make mistakes with the system at some time. That's why MiniMINC is designed to help prevent and correct errors. Nothing you type can ever damage it, so relax. *Book 3: MINC Programming Reference* explains the topic Error Messages in more detail; in addition, Appendix B of this book lists all error messages in alphabetical order, supplying with the message either a description of the error and its remedy or locations in the MiniMINC manuals where you can find that information.

RUNNING PROGRAMS WITH THE SYSTEM

Programs are sets of computer instructions or symbolic statements combined to perform tasks. They are put into motion by a command you type at the keyboard. Once a program begins to perform its task, it is said to be "running" or "executing." The types of programs that you can construct and run with MiniMINC are BASIC, graphic, and serial ASCII programs. The demonstration diskette in drive unit 0 contains examples of each type.

The Command Structure

Whenever READY and a blinking cursor appear on the screen, MiniMINC is ready to accept a command. A *command* is a word, mnemonic, or character which, by virtue of its syntax, causes a computer to perform a predefined operation. Command syntax requires a detailed explanation.

MiniMINC requires a particular format or syntax for its commands. By supplying the system with precise commands, you avoid needless repetition and ambiguity. Consider the RUN command as an example, but do not type it.

To execute the RUN command, you type the word RUN, a space, the program's name (FIRST), and press the RETURN key to bring this information to the system's attention.

RUN FIRST (RET)

MiniMINC checks the syntax of the command, locates the program named FIRST in its memory, and performs the steps represented collectively by the word RUN. However, if you type just R and the RETURN key, MiniMINC responds:

?MINC-F-Syntax error; cannot translate the statement

and typing RU evokes the same reply. The system comprehends the word RUN when you type it with just a RETURN, but executes whatever program currently resides in the MiniMINC terminal's *workspace* (a temporary holding area for data and programs in use), possibly performing a task you don't want.

Listing Directories

A telephone directory lists the occupants of a locality in alphabetical order and includes an address and phone number with each entry. A *directory* in MiniMINC lists the files of the same file type on a mass-storage volume in alphabetical order and includes the file's location and size. Every time you involve a diskette's files in an operation, MiniMINC consults the diskette's directory first. Remember, a file is a named collection of data situated on a diskette and can be either a program or a data file (such as a memo, letter, or table).

You can summon a directory with the DIRECTORY command. Type:

DIRECTORY (RET)

below the READY prompt. The names of the files on your demonstration diskette appear on the screen. *Book 2: MINC Programming Fundamentals* provides you with a closer look at the DIRECTORY command.

BASIC Programs

The MiniMINC programming language is MINC BASIC. Each command, statement, and function in MINC BASIC represents an integrated collection of computer operations. When you create a BASIC program, you are arranging a relatively small number of language elements to implement a far greater number of computer activities. In a way, you are directing MiniMINC's performance of your tasks with a form of shorthand.

BASIC Demonstration Programs The following two BASIC programs illustrate some fundamental problem solving and data manipulation techniques.

ALPHA, the first program, sorts names by alphabetical order and then displays them on the screen.

Should you encounter any problems using the program, consult Appendix A in the back of this manual for assistance. *Book 2: MINC Programming Fundamentals* explains many of the tech-

niques represented in ALPHA. *Book 3: MINC Programming Reference* contains descriptions of all MINC BASIC commands and statements arranged in alphabetical order.

Type:

RUN ALPHA (RET)

The program identifies itself, gives the current date and time, and then displays operating instructions. Type some names in the suggested format, including the word "end" as your last entry. ALPHA arranges the names alphabetically and prints them on the screen. The word READY indicates that the program finished and that MiniMINC is waiting for another command.

If you'd like to see the program, type:

LIST (RET)

Press the NO SCROLL key. This prevents the beginning lines of ALPHA from disappearing before you've had a chance to examine them. Press NO SCROLL again to resume scrolling.

If your MiniMINC equipment includes a printer, you can create a printed copy of ALPHA by typing:

COPY ALPHA LP: (RET)

TRIG, the other BASIC demonstration program, supplies the sine and cosine for any angle between 0 and 360 degrees. To begin, type:

RUN TRIG (RET)

TRIG identifies itself, briefly explains its purpose, and then requests your response. Type a "Y" to continue or "N" to end the program.

To see this program, type:

LIST (RET)

There's no need for NO SCROLL ; TRIG is so small that it fits completely on your screen.

If you have a printer, type:

COPY TRIG LP: (RET)

Graphic Programs

Programs that compute data and construct diagrams for the display of that data are called *graphic programs*. Graphic programs can also display data obtained from laboratory instruments. The common denominator of all graphic programs is their ability to represent data pictorially.

MiniMINC graphic programs use the MINC BASIC language, a form of standard Dartmouth BASIC modified to include such additional features as graphic routines. The MiniMINC user familiar with these routines can produce data displays that enable quick comprehension of important information.

Each graphic routine orchestrates MiniMINC's resources to achieve an effect or construct a particular framework for data display. For instance, there's a routine called BARGRAPH that formats data as a bargraph and another (BOX) that draws rectangular boxes. You supply the coordinates and modify the basic design by typing modifiers with the routine's name. You can also alter a graph after it appears on the screen.

A routine's name contains from one to three words separated by the underscore character (DISPLAY_CLEAR, for example). You can include it as an element of a program or use it separately to perform its task immediately after you type it. Either way, MiniMINC graphic routines simplify operations involving screen attributes.

Graphic Demonstration Programs The three graphic programs in this section demonstrate several display techniques possible with MiniMINC. If your experience running these programs varies from the following descriptions, consult Appendix A in this manual for assistance.

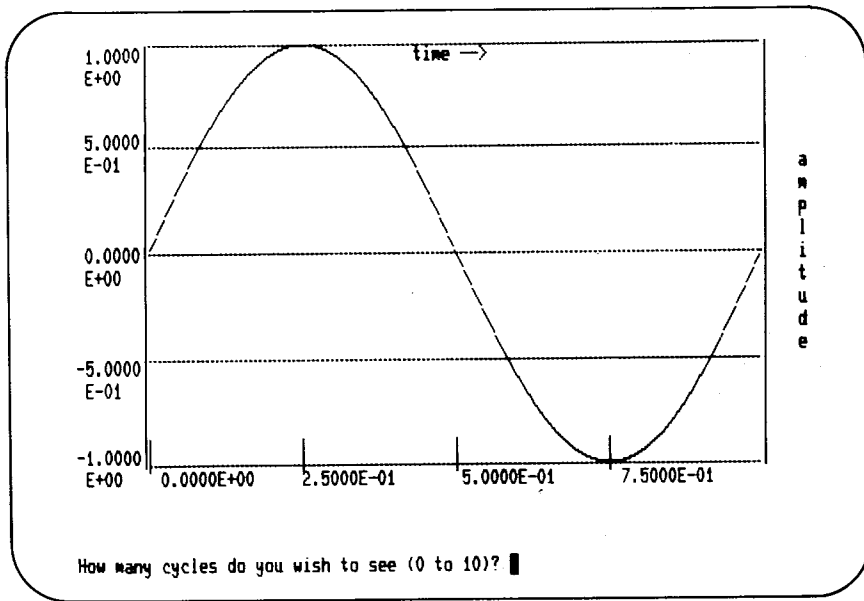
CYCLES displays one cycle of a sine wave and asks you what number of cycles to display as the next graph.

To run CYCLES, type:

```
RUN CYCLES (RET)
```

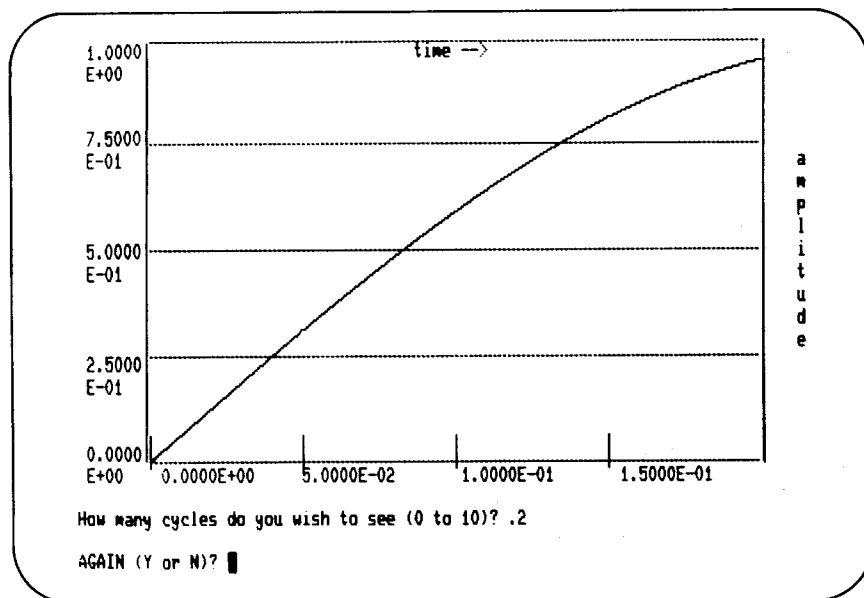
The program identifies itself, then creates a single-cycle display as shown in Figure 10. CYCLES' request for the number of cycles to display in the next graph appears near the screen's bottom with a flashing cursor. For the purpose of this demonstration, type:

The initial graph disappears, replaced by one displaying 2/10 of a cycle (see Figure 11). Notice that the coordinates automatically change to accommodate the smaller sample; this effect results from a MiniMINC feature called *autoscaling*, described in *Book 4:MINC Graphic Programming*.



MR-S-184-79

Figure 10. CYCLES — Single Sine Wave



MR-S-185-79

Figure 11. CYCLES — 2/10 Sine Wave

A request to repeat the procedure — AGAIN (Y or N)? — and a flashing cursor appear at the bottom of the screen. Type “Y” to repeat or “N” to end the program.

At the end of the exercise, you can examine the program by typing:

LIST (RET)

The graphic routines used in the CYCLES program are:

DISPLAY_CLEAR

GRAPH

LABEL

You will find descriptions of these routines in *Book 4: MINC Graphic Programming*.

If your MiniMINC equipment includes a printer, type:

COPY CYCLES LP: (RET)

The next program, APPROX, demonstrates a Fourier series approximation to a sawtooth function. First the program plots a sawtooth, making it disappear upon completion; then it plots the approximation points and recalls the sawtooth so that you can compare them.

The sawtooth function is:

$$f(x) = \begin{cases} x - \pi < x \leq \pi \end{cases}$$

The Fourier series approximation for $f(x)$ is:

$$f(x) \approx 2(\sin x - 1/2 \sin 2x + 1/3 \sin 3x - \dots)$$

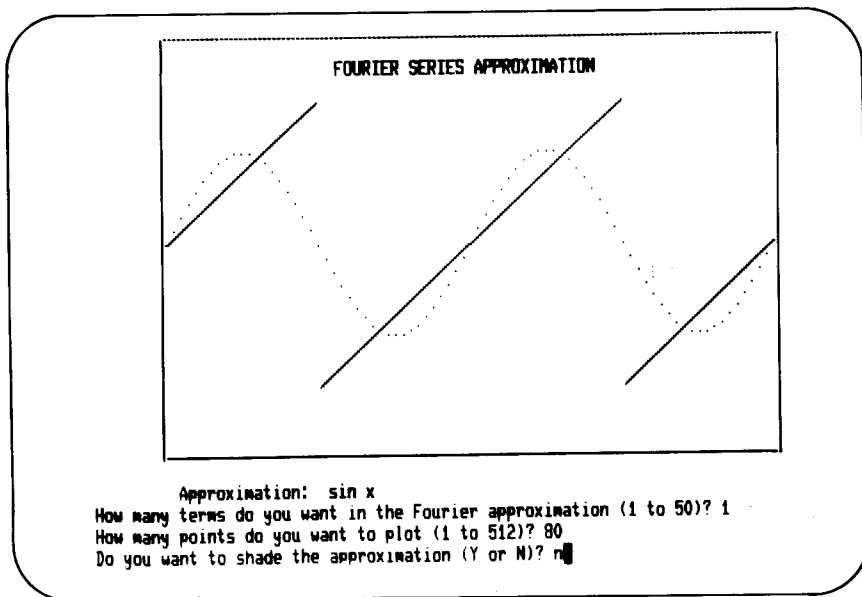
To run APPROX, type:

RUN APPROX (RET)

A sawtooth graph develops on the screen (see Figure 12). When completed, APPROX asks you for the number of terms you would like to include in the Fourier approximation, the number of points to plot, and if you want shading added to the graph for the approximation. For this discussion, type:

- 1 **(RET)** for the question on term number
- 80 **(RET)** for the question on point number
- N **(RET)** for the question on shading

A single-term approximation is simply a sine wave, a poor approximation to the sawtooth.



MR-S-186-79

Figure 12. APPROX — One-term Approximation

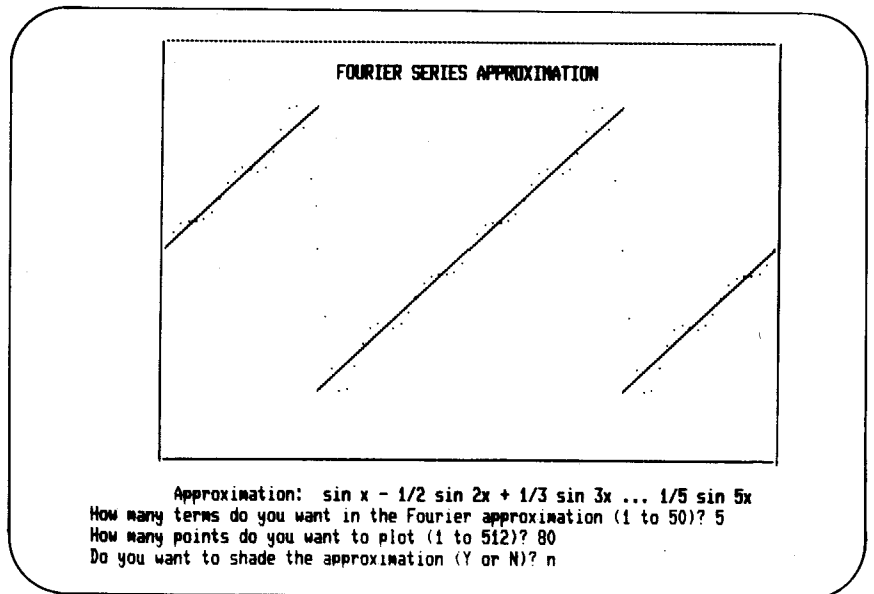
When APPROX asks if you want to run another case, type:

Y **(RET)**

Then, after APPROX plots another sawtooth, type:

- 5 **(RET)** for the question on term number
- 80 **(RET)** for the question on point number
- N **(RET)** for the question on shading

Note the five-term approximation's closer correspondence to the sawtooth (Figure 13).

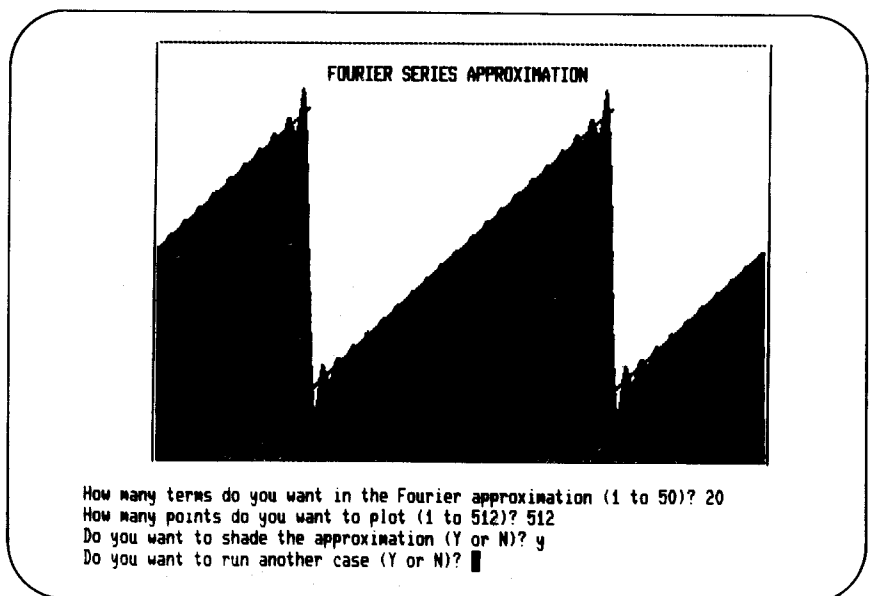


MR-S-188-79

Figure 13. APPROX — Five-term Approximation

Try a 20-term approximation (Figure 14), using shading this time to accentuate the curve's rapid rise and fall. Type:

Y **(RET)** for the question on running another case



MR-S-189-79

Figure 14. APPROX — Twenty-term Approximation

Then, after APPROX plots a fresh sawtooth, type:

20 (RET) for the question on term number

512 (RET) for the question on point number

Y (RET) for the question on shading

When you wish to end the program, respond with "N" to the repeat request.

The LIST and COPY commands operate as with the previous demonstration programs.

The graphic routines used in the APPROX program are:

DISPLAY_CLEAR

ERASE_GRAPH

ERASE_TEXT

GRID

HLINE

HTEXT

POINT

REGION

SHADE

VIEW

WINDOW

Book 4:MINC Graphic Programming contains full descriptions of these routines.

The final graphic program, WINDOW, demonstrates the simultaneous display of two separate graphs. WINDOW creates the graph of a damped sine wave in the screen's upper region and then displays a portion of that graph in the lower region.

Type:

RUN WINDOW (RET)

The program responds with a graph of the damped sine wave in the screen's upper region and asks what area of the graph you want expanded for the lower region.

Type:

10 (RET) for the first X coordinate request

20 (RET) for the second X coordinate request

Two vertical markers, called *brands*, appear on the graph, intersecting the damped sine wave at the X coordinates you specified. WINDOW then displays your selection expanded as a graph in the lower region with autoscaling adjusting the coordinates (see Figure 15). The program asks if you wish to try again, removing the brands and the lower graph if you type "Y". If you type "N", WINDOW terminates itself and the word READY appears as a sign of MiniMINC's willingness now to accept commands.

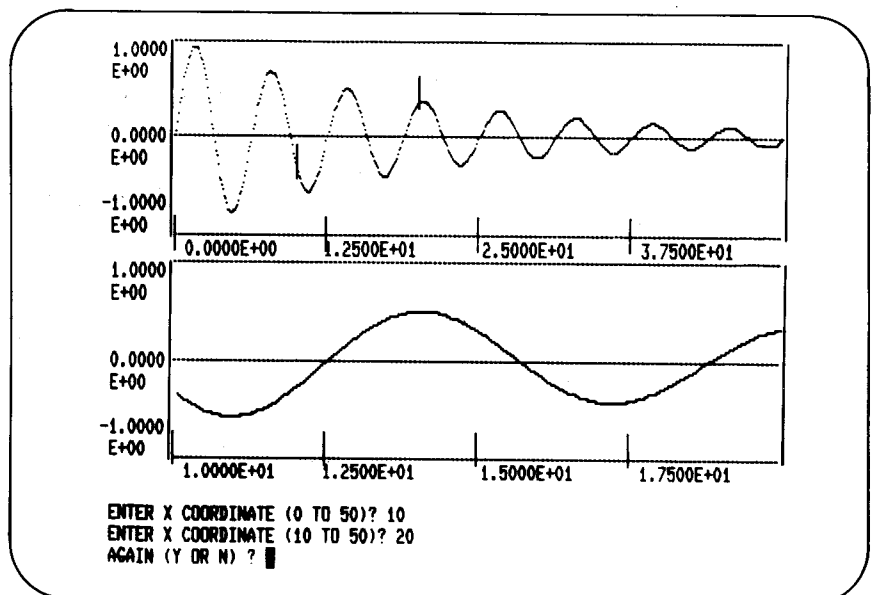


Figure 15. WINDOW Screen Sample

MR-S-191-79

Of course, the LIST and COPY commands perform as for the previous demonstration programs.

The graphic routines used in WINDOW are:

CHAR_MODE
 DISPLAY_CLEAR
 ERASE_GRAPH
 ERASE_TEXT
 GRAPH

HLINE
 HTEXT
 MAP_TO_TEXT
 MOVE_CURSOR
 POINT
 PUT_SYMBOL
 REGION
 ROLL_AREA
 WIDE_LINE
 WINDOW

You will find descriptions of these routines in *Book 4: MINC Graphic Programming*.

Few people can memorize the name and form of every command, statement, function, and routine in MiniMINC — and few would care to try. However, you can access this information in little more than the time required to reach for a MiniMINC manual or the *MiniMINC Summary*.

Getting Help

The HELP feature allows you to forego even the minor inconvenience of leafing through pages. You can summon lists of every command, function, statement, or routine to appear on your screen by typing the following commands:

HELP BASIC
 HELP CONTROL
 HELP FOURIER
 HELP GRAPHICS
 HELP SERIAL
 HELP SETUP

You can request a general description of these commands. Type:

HELP (RET)

The resulting text explains the contents of the six HELP category commands and describes how to display the form of each statement, function, and routine with HELP. The form descriptions resemble those contained in *Book 3: MINC Programming Reference*.

The HELP feature's form descriptions are located in a file on the user diskette. For this reason, a user diskette must be in drive unit 1 so that HELP can function. If you enter the HELP command when there is no user diskette in place, then the following error message appears:

```
?MON-F-Directory I/O error xxxxx
```

If you enter the HELP command and a user diskette is in drive unit 1, but it no longer has the HELP text file (some users erase the file to reclaim the space), the HELP feature displays the error message:

```
?HELP-W-Default system HELP file not found
```

```
no help for topic help
```

Every user diskette created from the Master User Diskette originally contains the HELP text file.

Creating HELP Files You can create your own HELP text files for use with the HELP feature. To construct one of these files, use the keypad editor described in *Book 2: MINC Programming Fundamentals*. The file can contain whatever information you desire, though the size of the file depends on the storage volume's available space.

Before you can use a personal HELP text file with the HELP feature, your file must meet the following conditions:

- The file's name must be from one to six characters in length (letters and digits only).
- The file's type must be TXT.
- The file must reside on a diskette in drive unit 1.

Once these conditions are satisfied, entering the HELP command and the file's name causes the file's contents to appear on the terminal screen.

For instance, you could create a personal HELP text file that explains the options available for the GRAPH routine. If you named this file GRAPH2.TXT and stored it on a diskette currently in drive unit 1, you could type:

```
HELP GRAPH2 (RET)
```

and the contents of GRAPH2.TXT would appear on the terminal screen.