MiniMINC

Book 1: Introduction to MiniMINC

August 1979

This document introduces the MiniMINC computer system, explains its features, and demonstrates its use. *Book 1: Introduction to MiniMINC,* which contains a programming primer, a system error message compendium, and a system glossary, addresses the new user in particular. The glossary and error message compendium will also benefit the experienced user.

Order Number AA-H400A-TC

MiniMINC

VERSION 1.1

digital equipment corporation • marlboro, massachusetts

First Printing, August 1979

DIGITAL	DECsystem-10	MASSBUS
DEC	DECtape	OMNIBUS
PDP	DIBOL	OS/8
DECUS	EDUSYSTEM	PHA
UNIBUS	FLIP CHIP	RSTS
COMPUTER LABS	FOCAL	RSX
COMTEX	INDAC	TYPESET-8
DDT	LAB-8	TYPESET-11
DECCOMM	DECSYSTEM-20	TMS-11
ASSIST-11	RTS-8	ITPS-10
MINC-11	DECSYSTEM-2020	MiniMINC

CONTENTS

FIGURES

PREFACE

This manual contains the following information:

- an overview of the MiniMINC computer system
- instructions for starting and using the system
- instructions for use of the demonstration programs ALPHA, APPROX, CYCLES, LISSJU, PLOT, TRIG, and WINDOW
- a programming primer for computer novices who wish to learn effective program development techniques
- an error message compendium for finding MiniMINC error messages with their descriptions and recommended actions, or references for finding this information in the manuals
- a system glossary for finding definitions of terms used in the MiniMINC manuals

CHAPTER 1
A DESCRIPTION OF MiniMINC

MiniMINC is a general-purpose, desktop computer system. With it you can perform calculations, develop programs, control experiments, monitor laboratory processes, and acquire data. MiniMINC's video graphic terminal displays your data in a manner you select from an assortment of modifiable graph forms.

MiniMINC is designed for convenient operation and installation.

The Equipment	runs on 115V ac or 230V ac in temperatures ranging from 15 C to 32 C (59 F to 90 F) and in a relative humidity of 20% to 80%.
Unpacking and Installing MiniMINC	tells you how to unpack, install, and test the system in straightforward, nontechnical terms.
The MiniMINC Manuals	• describe the system components • teach BASIC programming and MiniMINC system operation • explain MiniMINC's graphic capabilities • explain MiniMINC's serial ASCII, data processing, and program control capabilities • guide you through some helpful troubleshooting procedures

The MINC BASIC programming and command language	uses English-like commands and statements to promote quick comprehension.
The MINC Newsletter	keeps you current with programming hints, new applications, and maintenance bulletins. The MINC Newsletter is issued regularly to all MiniMINC owners.
The MINC Product Services Center	answers your MiniMINC-related questions through its toll-free phone service.

MiniMINC's large memory, graphics capability, and small size qualify it as a versatile desktop computer that's well suited for most computational needs. The system can communicate with external instruments such as plotters (for creating graphic representations on paper), printers (for producing paper copies of alphabetical, numerical, and special character data), and other instruments that conform to the EIA standard RS-232C for serial data transfer (see Chapter 4, Using MiniMINC With Serial ASCII Equipment).

You can interchange data between MiniMINC and another computer system with an optional package called MINC NFT (Network File Transfer). MINC NFT enables MiniMINC to share information and equipment with another Digital computer system when both systems are connected by modems (modulator/demodulator devices used in long distance data transmission), acoustic couplers, or direct cable.

THE MiniMINC EQUIPMENT

The MiniMINC equipment comprises the terminal, the keyboard, and the chassis pictured below.

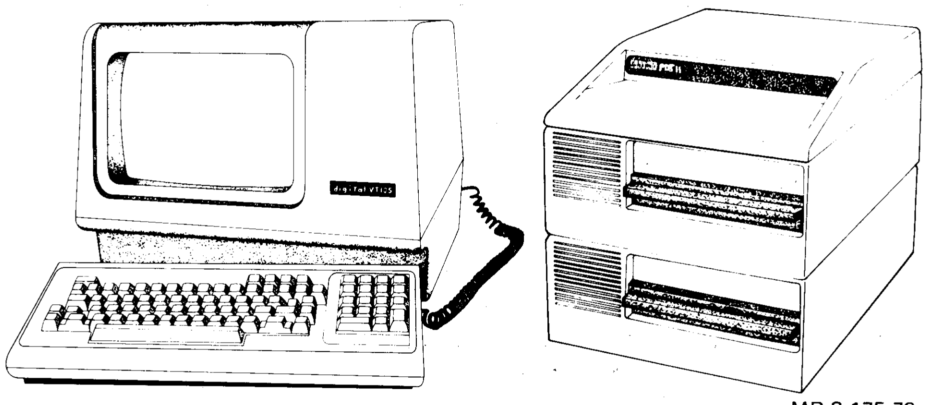


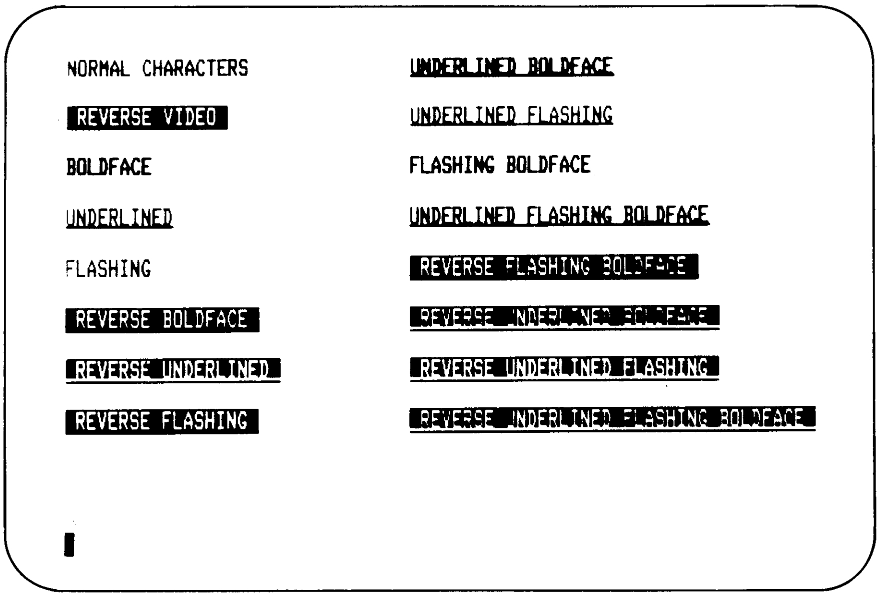
Figure 1. The MiniMINC Equipment

The Terminal

The MiniMINC terminal is your means of communicating with the system. MiniMINC receives most of its information from the keyboard and displays its queries, messages, and replies on the terminal's screen.

The terminal allows you to select a variety of display modes and character modes to suit your own taste or a certain environment. With the character modes you can display text in normal, boldface, flashing, underlined, or reverse video characters and can combine these modes in the same display. Screen display modes permit black or white backgrounds, 80- or 132-column widths, and smooth or jumping movement of text lines (scrolling). Your selections of character and display modes remain in effect until you change them explicitly.

Detailed descriptions of available modes exist in *Book 4: MINC Graphic Programming* (see CHAR_MODE and DISPLAY_MODE) and the *MiniMINC Supplement* (see Changing Operating Modes on the MiniMINC Terminal). Figure 2 shows some character mode combinations.



```
NORMAL CHARACTERS        UNDERLINED BOLDFACE
REVERSE VIDEO            UNDERLINED FLASHING
BOLDFACE                 FLASHING BOLDFACE
UNDERLINED               UNDERLINED FLASHING BOLDFACE
FLASHING                 REVERSE FLASHING BOLDFACE
REVERSE BOLDFACE         REVERSE UNDERLINED BOLDFACE
REVERSE UNDERLINED       REVERSE UNDERLINED FLASHING
REVERSE FLASHING         REVERSE UNDERLINED FLASHING BOLDFACE
```

MR-S-176-79

Figure 2. Some MiniMINC Character Modes

Supplied with the MiniMINC are over 30 graphic routines that arrange data in point-plot graphs, bargraphs, and strip charts for dynamic data representation (graphic routines can be used alone or within programs). Most of these routines accept modifications such as shading, an altered shadeline, and double-width characters. The terminal also can simultaneously display two graphs, decrease or increase the screen area devoted to text, and even find the graph coordinates of a point you select on the screen. Consult *Book 4: MINC Graphic Programming* and the *MiniMINC Supplement* for complete details and instructions on graphic programming. Figures 3 and 4 illustrate three of the many graphic variations possible with MiniMINC.

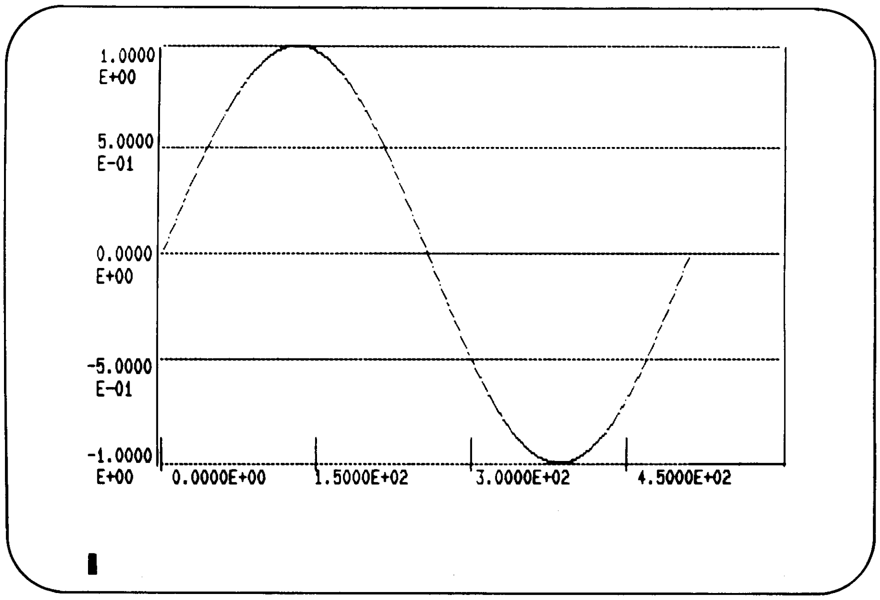


Figure 3. Sample Graph — Sine Wave

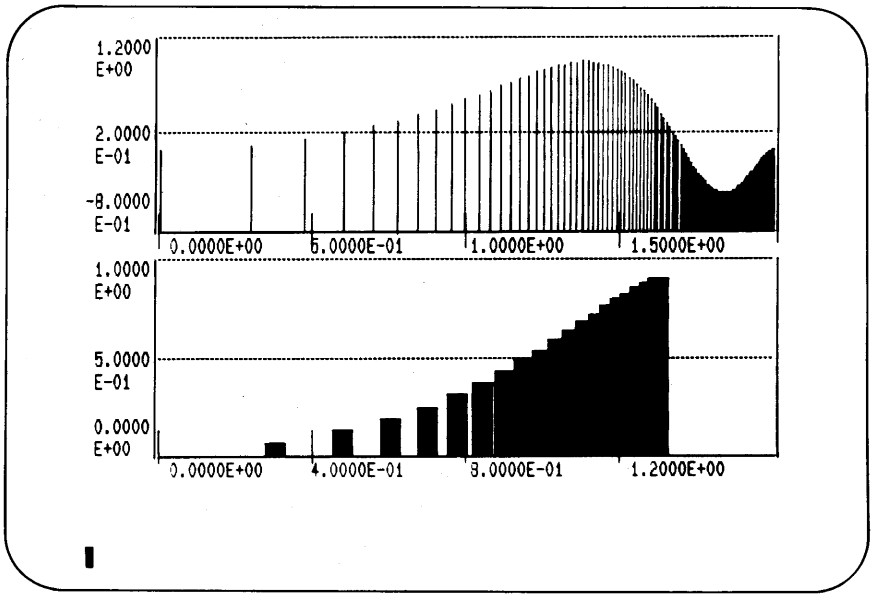


Figure 4. Split Screen with 2 Graphs — Bargraph and Point-plot Graph with Shading

The Keyboard

A detachable keyboard with 6 feet (1.8 m) of coiled cord accompanies the terminal (see Figure 1). Designed to resemble a standard typewriter keyboard, this unit also has an 18-key auxiliary keypad situated on its righthand side to facilitate rapid text manipulation (see Chapter 15, *Book 2: MINC Programming Fundamentals* for information on keypad editing). With the movable keyboard, your typing position can vary with your needs. This feature also permits others to view the terminal's screen while you and MiniMINC communicate. Chapter 2 contains an illustration of the keyboard and explains the function of some of its specialized keys.

The Chassis

A minicomputer and two diskette drive units are combined within a free-standing chassis that attaches to the terminal with a single cable.

The minicomputer stores, retrieves, and processes all MiniMINC instructions and data and regulates devices composing the system.

The chassis' two diskette drive units supply MiniMINC with a large program and data storage capacity. These drives accept one diskette each and retrieve or store information on them when you give the appropriate directive. Diskettes, described in the following section, are interchangeable, thus permitting information storage as extensive as your supply of diskettes.

On the back of the chassis are six ports or jacks that can accommodate serial line devices such as printers, graphic plotters, and many other instruments. These ports are referred to as SLU (Serial Line Unit) ports. A serial line transfers data from a sending to a receiving device according to a protocol established by EIA standard RS-232C for serial line data transmission (see Chapter 4 for more information on MiniMINC's serial lines).

DISKETTES

A diskette is a thin, flexible, oxide-coated disk similar in size to a 45-rpm phonograph record. A diskette is recorded on one side only and is permanently contained in an 8-inch square envelope.

A diskette is well suited as a storage and distribution medium. It's inexpensive, compact, and easily transportable via briefcase or manila envelope. Figure 5 shows the upper side of a diskette.

Random-access and *mass-storage* are two terms commonly used to describe the diskette and its drive unit. *Random-access* refers to the drive unit's ability to access data in any order. This form of

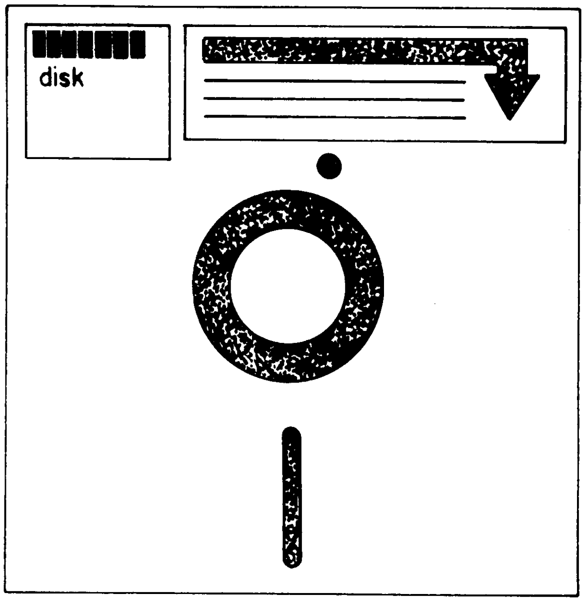


Figure 5. A Diskette

data acquisition contrasts with sequential-access devices like magnetic-tape and cassette drives that search for data in one direction only, sometimes winding through an entire tape and back again to find data. *Mass-storage* pertains to the large amounts of data a diskette can hold as contrasted to the computer's memory storage, which is more limited.

Each diskette contains 480 uniform divisions called *blocks*, and each block can hold 512 characters. A block is a standard unit of measure for diskettes used in MiniMINC to represent their occupied and unoccupied regions. Most characters are numeric (0-9), alphabetic (A-Z, lower and upper case), or special (@,!,$,etc.).

Collections of data (reports,tables,etc.) and programs are stored on diskettes as files. A file can occupy one or more adjacent blocks. When you direct MiniMINC to retrieve a file, you specify the file's own particular name as part of the direction (a command or statement) and MiniMINC searches the diskette for that file. The procedure requires a few seconds once you've typed the command. Chapter 3, Using MiniMINC, describes the structure and use of commands in detail.

THE MiniMINC PROGRAMMING SYSTEM

So far, this manual has focused on the equipment, that is, the physical portion of MiniMINC. Yet the equipment, while important, constitutes only one part of the MiniMINC computer system.

The programming system is an intangible part of MiniMINC.

It's composed of system programs each of which, like any other program, is a set of computer instructions combined to perform certain tasks. Unlike most programs, including the demonstration programs you will be using in Chapter 3 of this manual (or the user programs that you might create later on), system programs are essential for the operation of the MiniMINC system. They initiate, coordinate, and monitor the equipment functions necessary for executing your commands, statements, and user programs. System programs operate the system.

As a user of MiniMINC, you have in your possession a *system diskette*. The system programs comprising the MiniMINC programming system are on this diskette; they can perform their tasks only after you have loaded the system diskette into its designated drive and properly initiated operation (Chapter 2, Starting MiniMINC, contains all the required steps).

Invisibility shields the system's programs from inspection or alteration. If you command MiniMINC to list the name of every file on the system diskette (remember that a diskette has a total capacity of 480 blocks), the number of free and used blocks listed for a new system diskette will appear to equal less than 100. This apparent incongruity exists because the large number of blocks used by the system programs is shielded from common use and thus protected from deletion or interference. However, you can still use the free blocks on the system diskette for personal data and program storage.

User diskettes (diskettes that store user programs) contain a system file that supports the HELP feature (see Chapter 3, Getting HELP).

Up to this point you've encountered the terms *program*, *routine*, *file*, and *data*, and perhaps experienced some confusion over their precise meanings. Consider the following definitions:

program	a set of computer instructions or symbolic statements combined to perform some task. A program can be part of the system or written by a user. It can contain multiple routines.
routine	a set of instructions arranged in proper sequence to cause a computer to perform a desired operation. Some routines are an integral, predefined part of the system.
file	a collection of data treated as a unit, which occupies one or more blocks on a mass- storage

	volume, and has an associated file name and file type.
data	a term used to denote any or all facts, numbers, letters, and symbols: basic elements of information that can be processed by a computer.

A *system program* is a set of computer instructions written in an assembly language (a symbolic programming language commonly used in the development of programming systems) and translated into a machine language (the actual language used by the computer when performing operations). A *user* (or *demonstration*) *program* is a set of symbolic statements frequently written in a high-level language like MINC BASIC or FORTRAN that triggers a sequence of computer operations with each statement or command. Thus, in a general way, a language is a machine language when one instruction corresponds to a computer operation and a high-level language when one instruction (or statement) corresponds to a sequence of computer operations.

For instance, when combined with a program name, the MINC BASIC command RUN causes a search for the program on the diskette, the conveying of its image into memory, a scan of the program for consistency, preparation of the computer's internal state, and finally, execution of the designated program.

A MiniMINC *routine* is a set of instructions invoked by a single statement that causes the computer to perform an often-repeated set of operations. MiniMINC routines come as part of the system.

A *file* can contain a program or a collection of data such as a report, the results from an experiment, or a statistical compilation.

To create a file, all you need is sufficient space on your diskette, a unique file name and type to identify it, and the appropriate command. Entering a program into MiniMINC requires standard typing procedure; entering data requires program interaction with external instruments or the MiniMINC editor, a system facility that allows you to create, modify, or inspect any text file (Chapter 15 of *Book 2: MINC Programming Fundamentals* fully describes the editor and its use). It is important to remember that the term *file* refers to any gathering of information stored on a diskette, not just facts and figures but programs as well.

The MINC BASIC Language

MINC BASIC is a high-level language developed for MINC (Modular INstrument Computer), MiniMINC's larger laboratory-oriented cousin. An offshoot of the standard BASIC language, MINC BASIC serves as a medium for both operating the computer and programming. Since many other computer systems require a set of commands separate from the programming language for their operation, MiniMINC's use of a common programming and operating language stands out.

MINC BASIC has concise commands and statements that resemble English but avoid its ambiguities. For example, the EDIT command invokes the editing feature described earlier for modifying data or program files and the INITIALIZE command performs a sequence of steps that prepares a new diskette for use with MiniMINC. This matching of command name with function facilitates quick comprehension and ease of use. *Book 2: MINC Programming Fundamentals* describes and illustrates the fundamentals of MINC BASIC while *Book 3: MINC Programming Reference* presents each command and statement in alphabetical order with a complete description of each.

MiniMINC Graphic Routines

When you invoke a graphic routine by typing its one- or two-word name, you can summon features such as simultaneous display of two graphs on the terminal screen, bargraphs, dynamic strip chart modes, and many more. *Book 4: MINC Graphic Programming* and the *MiniMINC Supplement* together describe all the routines and how to use them.

MiniMINC Serial ASCII Routines

The MiniMINC system can communicate with serial ASCII (American Standard Code for Information Interchange) equipment through the serial ASCII routines. Incorporated within a program or used alone, these routines allow MiniMINC to transmit and receive data over serial lines connected to the SLU ports on the back of the chassis. Chapter 4 and the *MiniMINC Supplement* provide additional information on the routines.

MiniMINC Data Processing Routines

The MiniMINC system includes two data processing routines, FFT (Perform Fast Fourier Transform) and POWER (Calculate Power Spectrum Coefficients). FFT creates a discrete, numerical approximation of a continuous Fourier transform based upon a set of data. POWER calculates the power spectrum of a set of data by using the real and imaginary coefficients calculated by FFT. The *MiniMINC Supplement* explains these data processing routines in detail.

MiniMINC Program Control Routines

MiniMINC's program control routines permit programs to suspend execution for a specified time interval and to perform a designated procedure when a specified time event occurs. The time event is either completion of a specified time interval or a specified time of day. The *MiniMINC Supplement* provides complete descriptions of these routines.

MiniMINC DOCUMENTATION

MiniMINC's manuals are a major reason for the system's ease of use. They explain and expand on such topics as:

- MiniMINC's capabilities
- MiniMINC's operation
- MINC BASIC programming
- MiniMINC's routines
- diagnosis and correction of possible MiniMINC problems

Unpacking and Installing MiniMINC tells you how to unpack and assemble the equipment in clear language accompanied by uncomplicated diagrams. You should have found this document attached to the outside of one of MiniMINC's shipping cartons.

Book 1: Introduction to MiniMINC acquaints you with the MiniMINC system. This manual familiarizes you with the concepts and terminology relevant to MiniMINC, its components and operation, and the perspective necessary to program effectively. The back of the manual contains directions for reading the other MiniMINC manuals, a glossary of computer terms used in the MiniMINC manuals, and a compendium of system error messages. This document is directed particularly at the computer novice, although the error message compendium benefits all MiniMINC users.

Book 2: MINC Programming Fundamentals leads you through the procedures for calculating, programming, and editing with MINC BASIC. Each chapter is designed to increase your comprehension and familiarity with MINC BASIC over that gained from the previous chapter. It provides numerous programming examples that you can try with MiniMINC while you read. This manual continues where the Introduction to MiniMINC leaves off.

Book 3: MINC Programming Reference is an alphabetically-ordered reference book containing detailed descriptions of every MINC BASIC command and statement. Book 3 also discusses such topics as Routines and Error Recovery. This manual gives examples and cross-references to Book 2 when applicable.

Book 4: MINC Graphic Programming teaches the concepts of graphic programming for the MiniMINC system. The back of the manual contains descriptions of the graphic routines arranged in alphabetical order and fortified with examples and illustrations. You should read Books 1 through 3 before starting this manual.

Note that Books 2 through 4 are also part of the MINC manual set, due to MINC and MiniMINC sharing the MINC BASIC language. For this reason, the books contain references to MINC equipment and manuals. You should ignore these references or, in some cases, substitute the appropriate MiniMINC term.

The *MiniMINC Supplement* contains such things as detailed troubleshooting procedures, directions for running the terminal's self-test and setting terminal characteristics, and information on optional devices such as printers that are available with the MiniMINC system.

The *MiniMINC Summary* presents the format and a one-line description of each MINC BASIC command, statement, and routine. It serves as a pocket-ready reference, something you can use at the terminal to recall a command's name, function, or form.

DEMONSTRATION PROGRAMS

Demonstration programs are comparable to those programs you will create for your own particular applications. In fact, some of the demonstration programs used in Book 2 perform tasks fundamental to many real-life operations.

The MiniMINC manuals make liberal use of demonstration programs as interactive aids to learning MINC BASIC and the routines. Some you type in as you progress with a lesson; others already reside on the Demonstration diskette. If you follow the directions and fully participate in the computer exercises, you should find MiniMINC an extremely easy system to learn.

CHAPTER 2
STARTING MiniMINC

This chapter explains the proper handling of MiniMINC equipment and the preparations required for using the system.

POWERING THE SYSTEM

The terminal and chassis each have a power switch located on their back panels (see Figure 6). Turn on the terminal by pulling up on the switch and allow the terminal ten to twenty seconds to warm up. Some of its lights will flash as it tests itself, then it will "beep" when finished. Proceed when a small blinking box or underscore (called a cursor) appears in the screen's upper left corner. If the screen remains blank, examine the terminal's back panel and make certain that the power cord is connected to both the terminal and the power source.

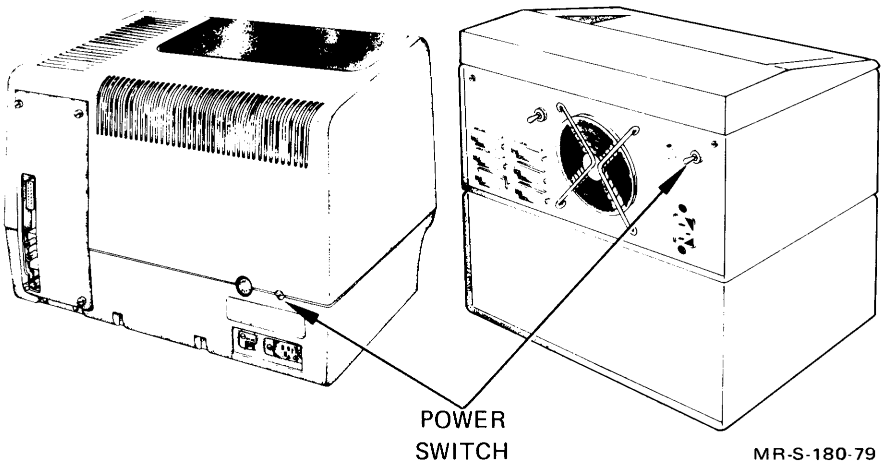


Figure 6. MiniMINC Switch Settings

Before turning on the chassis, open both of the drive doors located on its front panel and remove any diskettes (the next section describes proper handling of diskettes). A drive door opens when you squeeze the middle of its central ridge between your fingers. Release it, and the door lifts under its own power.

Now turn on the chassis. When you pull up on the power switch, three red indicator lamps situated in the chassis' top light up; they are labeled 1, 2, and POWER. Lamp 1 should go out immediately, leaving POWER and 2 burning steadily. RUN, the fourth indicator lamp, lights up later and remains lit while the MiniMINC programming system is running. Figure 7 shows the placement of these lamps. Should the appropriate lamps fail to light, refer to the Troubleshooting section of the *MiniMINC Supplement.*

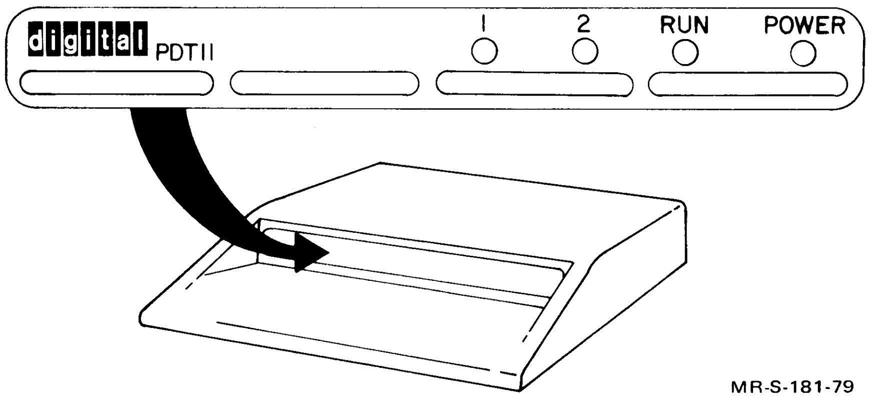


Figure 7. Light Panel on the MiniMINC Chassis

Also on the chassis' back panel is the mode switch, a three-way toggle switch marked RESET, NORMAL, and TEST (see page 37). RESET, the momentary up position, prepares the chassis for acceptance of the MiniMINC system programs; you should make a habit of doing one or two RESETs before inserting the system diskette or you may damage the MiniMINC programs it contains. NORMAL is the position to which the toggle automatically returns after a RESET; this position allows normal operation of the chassis. TEST, the down position, starts the self-tests built into the chassis and should be avoided during normal system operations; the *MiniMINC Supplement* contains all directions regarding test procedures. Any attempt to use MiniMINC with the mode switch in the TEST position may erase the contents of diskettes inside the chassis.

Move the mode switch to RESET. The toggle automatically returns to NORMAL and the indicator lamp 1 comes on, then goes out. Lamp 2 comes on. The chassis is now ready to accept the system diskette.

HANDLING DISKETTES

Diskettes require special care to prolong their usefulness and the accuracy of their contents. Like anything else in this world, diskettes wear out; however, you can prolong their life with the following precautions:

- Avoid touching the internal disk. Handle only the plastic envelope surrounding it.
- Never write directly on the diskette's envelope with anything but a felt-tip pen. For identification purposes, attach previously marked labels.
- Remove unnecessary labels when possible. Excess labels can interfere with the loading of a diskette.
- Do not apply chemicals or solutions to the envelope or internal diskette.
- Do not attach rubber bands or clips to the diskette.
- Avoid placing objects on the diskette.
- Keep diskettes away from sharp objects and magnetic fields (such as the one generated by the MiniMINC terminal).
- Do not bend or mutilate.
- Never remove the internal disk from its plastic envelope.
- Keep diskettes away from dust and dirt.
- Keep from direct sunlight or excessive heat.
- Allow diskettes to reach room temperature before using them.

Neglect of diskettes can cause loss of information and unnecessary expense. Protect them.

To load a diskette, gently insert it into an empty drive unit as shown in Figure 8. Notice that the label enters last and face up.

When the diskette is fully inserted, push down the ridge on the drive door until you hear a click. The diskette is now loaded.

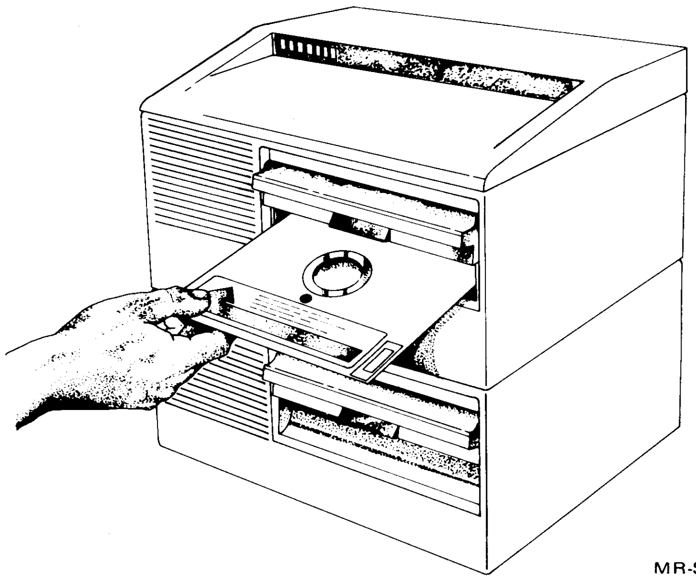


Figure 8. Inserting a Diskette

Diskettes should be returned immediately to their protective pockets after removal from the drive unit.

THE DISKETTES SUPPLIED WITH MiniMINC

Examine the diskettes you have at hand. If you are the first to use this MiniMINC system and you possess all the diskettes provided with MiniMINC, then you have seven diskettes:

1 Master Demonstration Diskette

1 Master System Diskette

1 Master User Diskette

3 blank diskettes

1 System Exerciser Diskette

Master diskettes cannot be erased or otherwise altered by conventional methods like regular diskettes. The master diskettes' sole function is to generate system, demonstration, and user diskettes. Store them safely after use.

The Master Demonstration Diskette generates demonstration diskettes. Demonstration diskettes contain a complete

MiniMINC programming system, together with the demonstration programs required for exercises in this manual.

The Master System Diskette generates system diskettes that contain the MiniMINC programming system and nothing else.

The Master User Diskette prepares (initializes) a blank diskette for the eventual storage of programs and data files. The resulting user diskette also contains a file used by the MiniMINC HELP feature (see Getting HELP in this manual).

Store the System Exerciser Diskette in a safe place. DIGITAL Field Service personnel use it to locate problems should you ever need on-site repairs. If you want to run the System Exerciser yourself, refer to Appendix A of the *MiniMINC Supplement* for operating instructions.

Treat the System Exerciser diskette as you would a master diskette by copying the original diskette, storing the original in a safe place, and then using the copy to exercise the system. This precaution can save you from inadvertently erasing your only copy.

KEYBOARD CHARACTERISTICS

Figure 9 shows the terminal keyboard. The arrangement of the large, left-hand group of 65 keys closely resembles a standard typewriter keyboard. The unfamiliar keys are called special terminal keys. This section describes only the special terminal key functions necessary for performing the demonstration programs used in this manual. *Book 2: MINC Programming Fundamentals* supplies details on all special terminal keys.

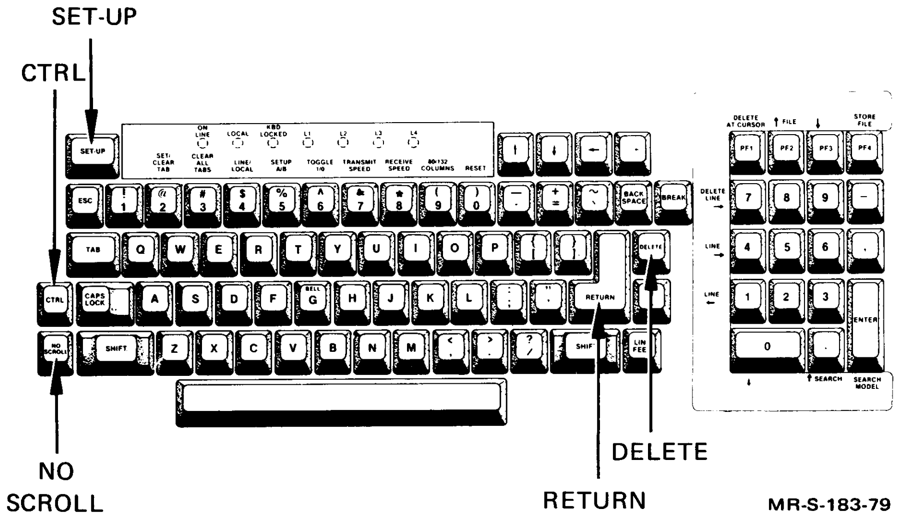


Figure 9. The MiniMINC Keyboard

Book 2: MINC Programming Fundamentals also describes the text-editing procedures reserved for the the small, right-hand group of keys and the arrow keys located on the main keyboard.

The SET-UP key is located in the upper left-hand corner of the main keyboard (see Figure 9). This key plays an important part in the alteration of screen and terminal attributes, as described in *Book 4: MINC Graphic Programming* and the *MiniMINC Supplement*.

The CTRL key, located on the keyboard's left side (see Figure 9), works in tandem with several letter keys to perform control commands. *Control commands* interrupt user and computer operations to assert your "control" over the computer. For instance, the demonstration programs in this manual use CTRL/C, a common control command that halts program execution and MiniMINC operations immediately. To invoke CTRL/C, simply press the letter "C" key (lower- or upper-case) while holding down the CTRL key; interruption of some MiniMINC operations requires two strokes of the "C" key while holding down the CTRL key. You implement all control commands in this fashion, merely substituting the appropriate letter for the "C" in this example. Books 2 and 3 describe the control commands and their functions.

The NO SCROLL key, situated below the CTRL key, halts the continuous display of information (even graphic displays) on the terminal screen. Whenever the terminal must display more characters than its screen capacity of 24 lines (with 80 or 132 characters per line), MiniMINC makes room for a new line of information by erasing the top line of the current contents of the screen, moving the remaining 23 lines up, and writing the new line at the bottom. Sometimes this scrolling occurs faster than you can read. At such a time, you can press the NO SCROLL key and the scrolling halts, resuming when you press the NO SCROLL key again.

The RETURN key, located on the right side of the main keyboard (see Figure 9), terminates most commands and statements (except control commands). Until you press this key, MiniMINC ignores whatever you type; press it, and MiniMINC accepts the typed line. This feature permits you the luxury of considering what you've typed before you submit it to MiniMINC for analysis and execution. Throughout this manual, one press of the RETURN is represented by (RET) .

The DELETE key, placed next to the RETURN key, erases typing mistakes. Each time you press the DELETE key, the most recent character disappears. In this fashion, you can erase backward to an earlier mistake and then retype fresh characters. You cannot, however, use the DELETE key to erase a character on a preceding line. To change a previous line, use the keypad editor, following the procedure in *Book 2: MINC Programming Fundamentals*. For changing lines in programs, see the description of the SUB command in *Book 3: MINC Programming Reference*.

You can repeat any letter, number, or special character by holding down its key (except TAB, RETURN, and CTRL). The character will reproduce for as long as you hold down the key or until it encounters the end of the current line, whichever is first. Even the DELETE key repeats when you hold it down.

ENGAGING THE SYSTEM

You should now be ready to start the MiniMINC programming system. The terminal and chassis are powered, the chassis' mode switch is in the NORMAL position (following a RESET), and you're familiar with the MiniMINC diskettes and their proper handling and loading.

If you have a diskette labeled Demonstration Diskette, then someone has already created the diskette you need for the remainder of this manual. You don't need to generate another one with the Master Demonstration Diskette. Instead, load the demonstration diskette into the chassis' top drive unit (called unit 0).

If you do not have a demonstration diskette, take the Master Demonstration Diskette, load it into drive unit 0, and proceed directly to the next section, Using the Master Diskettes.

Now, assuming that you have a demonstration diskette in drive unit 0, type two @'s in succession. As on a typewriter, typing an @ requires use of the SHIFT key. The indicator lamp 2 goes out and the RUN lamp comes on. Don't be alarmed by the chassis' clunking noises. You should expect these sounds of the drive heads seeking information on the diskettes with almost every request you make of MiniMINC.

USING THE MASTER DISKETTES

Master diskettes perform a single function: to transform a blank or recycled diskette into a system, demonstration, or user diskette. Store your masters in a safe place so that you can always create copies.

The next few paragraphs describe how to use masters. If you currently have a demonstration diskette in drive unit 0, read this section for future reference but do not act upon it. Reserve further action for the next section, Entering The Date And Time.

To create a demonstration, system, or user diskette, insert the appropriate master in drive unit 0 after powering up MiniMINC and pushing the mode switch to the RESET position. Now insert a blank diskette in drive unit 1, the bottom slot in the chassis' front panel. Hold down the SHIFT key and press the @ key twice. Instructions will appear on the screen.

During the duplication process, MiniMINC asks you for permission to proceed, reports its progress, and issues instructions.

Part way through the instructions, MiniMINC asks for the blank diskette's new *volume identifier* (Vol id) and new *owner name* (Owner name). These names are assigned to the diskette and appear whenever you request the diskette's directory. The volume identifier can be any identifying name; for instance, you might wish to identify a demonstration diskette as MINIMNC DEMO. The owner name can be anything except DIGITAL. The volume identifier and owner name must each be 12 characters or less in length (including spaces), otherwise MiniMINC truncates them at 12 characters.

When the duplication process is finished, remove the duplicate diskette from drive unit 1 and label it appropriately (demonstration diskette, system diskette, or user diskette). Remove the master from drive unit 0 and store it safely.

If you created a demonstration diskette to keep pace with this manual, push the chassis' mode switch to the RESET position, install the new demonstration diskette in drive unit 0, and press the @ key twice while holding down the SHIFT key.

ENTERING THE DATE AND TIME

Typing the date and time for MiniMINC is a necessary part of the start-up procedure. Once you provide these values, MiniMINC's system clock and calendar take over maintaining the date and time until you turn off the system.

An accurate date and time can help you in several ways. MiniMINC affixes the date to every file you transfer, alter, or create. This information allows you to easily distinguish your most recent entries from older files, thus saving you the necessity of scanning several files to find your last stopping point. MiniMINC also has functions that supply the time whenever executed; when included in your programs, they can trigger time-related events.

When MiniMINC displays an identifying message and a request for the date, type the day, month, and year as follows:

dd-mmm-yy (RET)

where:

dd	is the day of the month
mmm	is the first three letters of the month's name
yy	is the last two digits of the year
(RET)	represents one press of the RETURN key.

NOTE

This manual represents everything you should type at the terminal in red. Enter all characters and spaces exactly as shown.

Example:

```
MiniMINC V1.1

Please enter
Today's date: 5-JAN-79 (RET)
```

Remember, use the DELETE key to correct typing mistakes.

After you press the RETURN key, MiniMINC responds with:

```
current time:
```

To which you reply:

hh:mm:ss (RET)

where:

hh	is the hour of the day in 24-hour notation
mm	is the minute

ss	is the second
(RET)	represents one press of the RETURN key.

For 24-hour time notation, add 12 to anything from 1 P.M. to midnight. For instance, 10 A.M. remains 10 but 1 P.M. becomes 13. Midnight is 0. For example,

```
MiniMINC V1.1

Please enter
Today's date: 5-JAN-79
current time: 15:32:00
```

MiniMINC responds with the word READY, signaling the system's readiness to accept a command or statement (the same is true of a system diskette).

STARTING MiniMINC — A SUMMARY

The general procedure for starting MiniMINC is as follows:

1. Check the drive units and remove any diskettes.
2. Turn on the terminal and the chassis.
3. Push the chassis' mode switch to the RESET position.
4. Install a system or demonstration diskette in drive unit 0.
5. Press the @ key twice while holding down the SHIFT key.
6. Enter the date and time.

The word READY appears, signaling MiniMINC's readiness to accept a command or statement.

CHAPTER 3
USING MiniMINC

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

CALCULATING WITH MiniMINC

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.

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 techniques 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.

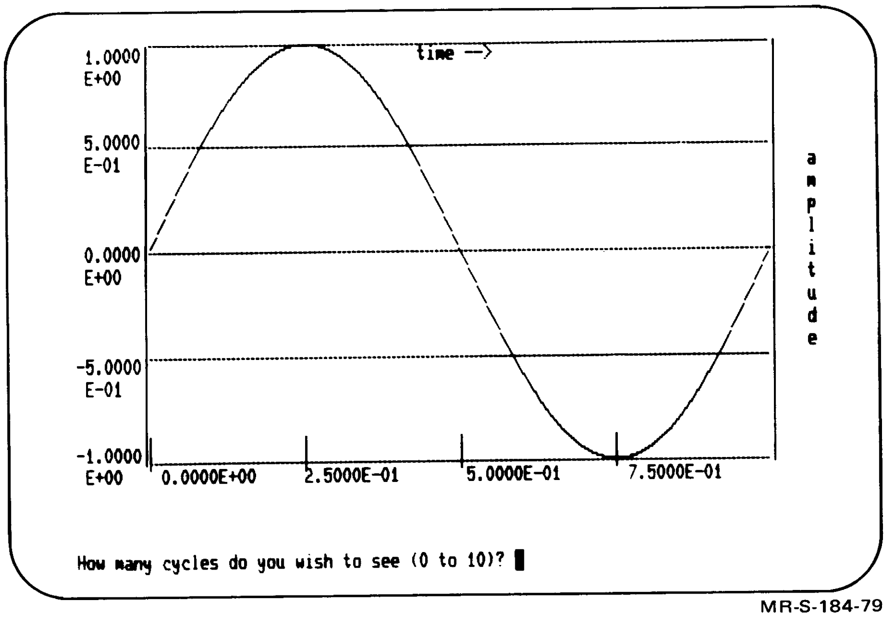
To run CYCLES, type:

```
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:

```
.2 (RET)
```

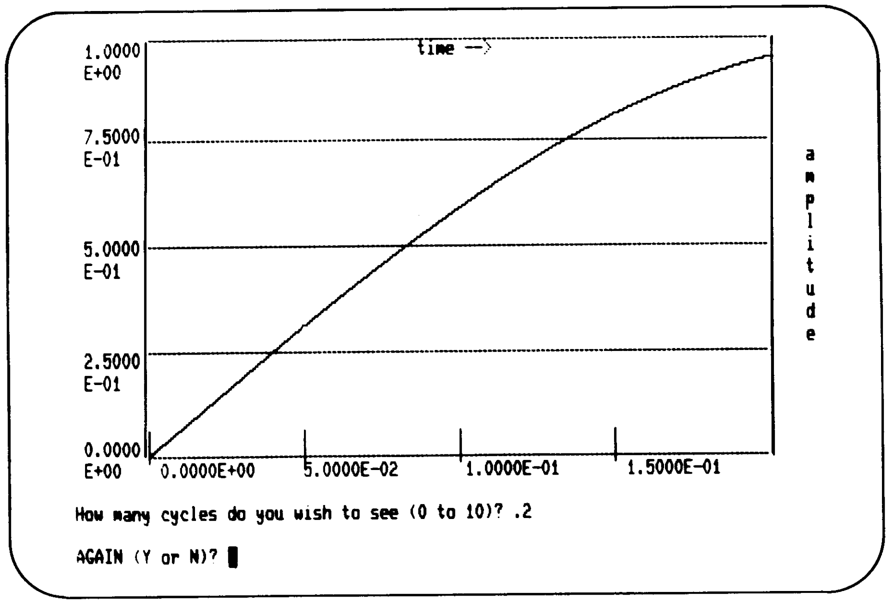
The initial graph disappears, replaced by one displaying 2/10 of a cycle (see Figure 11). Notice that the coordinates automatically change to accommodate the smaller sample; this effect results from a MiniMINC feature called *autoscaling*, described in *Book 4:MINC Graphic Programming.*



```
How many cycles do you wish to see (0 to 10)? ▌
```

MR-S-184-79

Figure 10. CYCLES — Single Sine Wave



```
How many cycles do you wish to see (0 to 10)? .2
AGAIN (Y or N)? ▌
```

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:

$f(x) = X \quad -\pi \leqslant X \leqslant \pi$

The Fourier series approximation for f(x) is:

$f(x) = 2(\sin x - 1/2 \sin 2x + 1/3 \sin 3x - \ldots)$

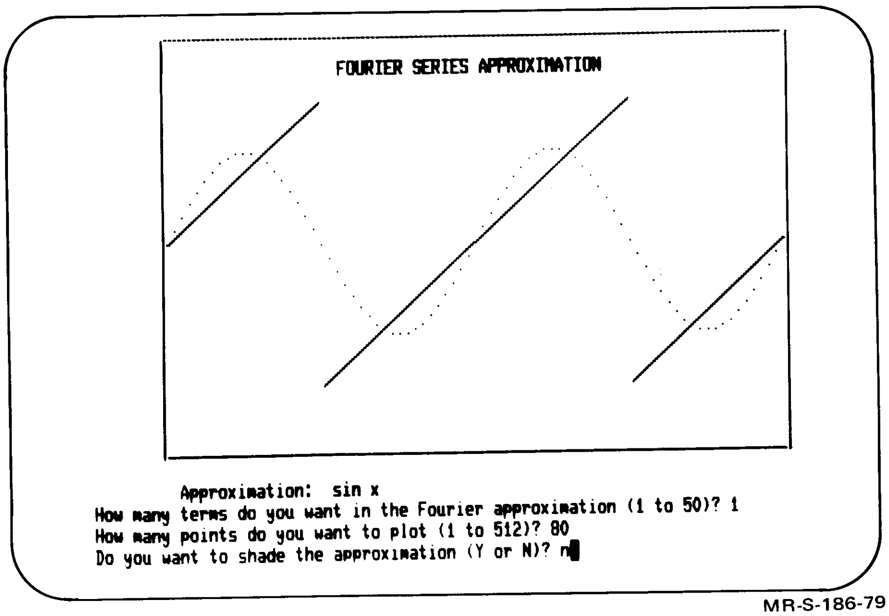
To run APPROX, type:

```
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.



```
Approximation: sin x
How many terms do you want in the Fourier approximation (1 to 50)? 1
How many points do you want to plot (1 to 512)? 80
Do you want to shade the approximation (Y or N)? n
```

MR-S-186-79

Figure 12. APPROX — One-term Approximation

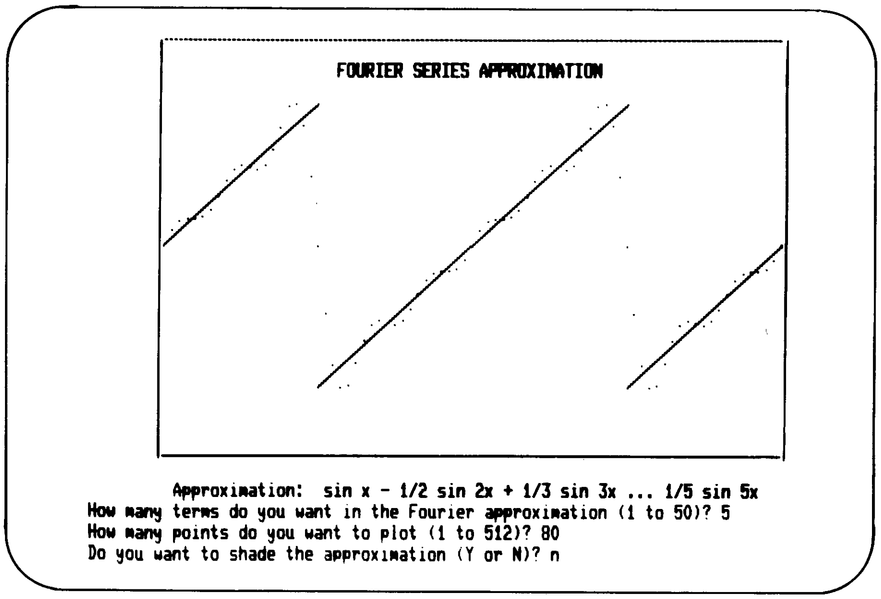
When APPROX asks if you want to run another case, type:

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).



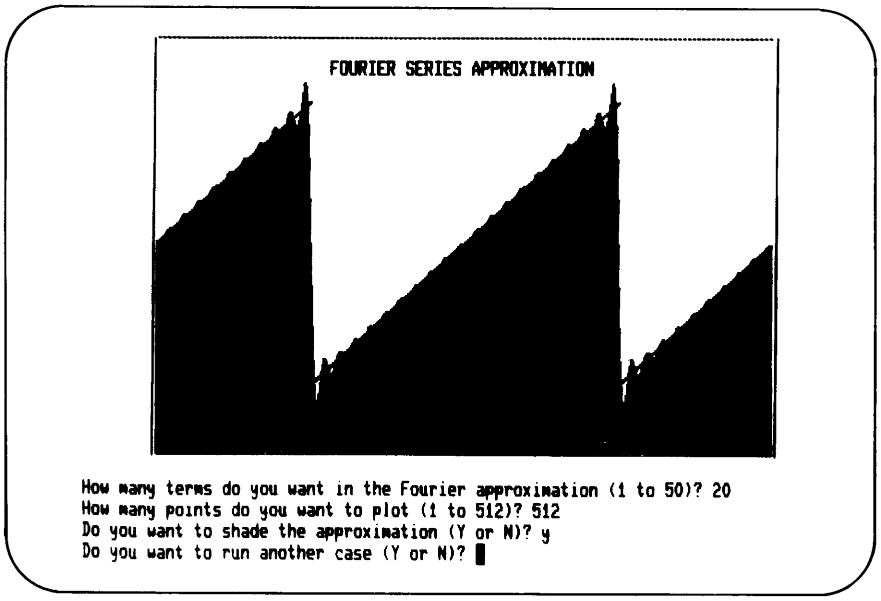
```
        Approximation:  sin x - 1/2 sin 2x + 1/3 sin 3x ... 1/5 sin 5x
How many terms do you want in the Fourier approximation (1 to 50)? 5
How many points do you want to plot (1 to 512)? 80
Do you want to shade the approximation (Y or N)? n
```

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



```
How many terms do you want in the Fourier approximation (1 to 50)? 20
How many points do you want to plot (1 to 512)? 512
Do you want to shade the approximation (Y or N)? y
Do you want to run another case (Y or N)? █
```

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:

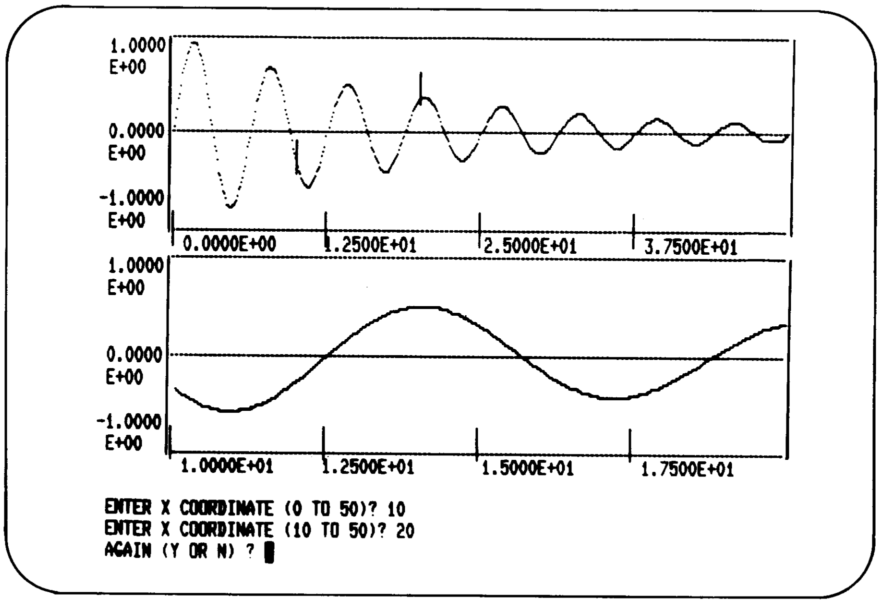
```
RUN WINDOW (RET)
```

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.



```
ENTER X COORDINATE (0 TO 50)? 10
ENTER X COORDINATE (10 TO 50)? 20
AGAIN (Y OR N) ? ▮
```

Figure 15. WINDOW Screen Sample

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.*

Getting Help

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*.

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.

CHAPTER 4
USING MiniMINC WITH SERIAL ASCII EQUIPMENT

This chapter gives a general description of how MiniMINC communicates with serial ASCII equipment and describes how to use LISSJU.BAS, a demonstration program that creates Lissajous figures on an interactive plotter.

SERIAL ASCII EQUIPMENT

MiniMINC can communicate with instruments that transmit characters according to EIA standard RS-232C. This standard sets the protocol for serial transfer by which most terminals (including the MiniMINC terminal) and some other instruments, such as interactive plotters and analytical balances, transmit and receive ASCII characters (see Appendix A in *Book 2: MINC Programming Fundamentals* for a table of ASCII codes). *Serial transfer* is a method of data transmission in which the components of a data word are transmitted one after another (serially) along a single pair of lines from a sending to a receiving device.

MiniMINC's SLU PORTS

You connect devices to MiniMINC via the six SLU (Serial Line Unit) ports located on the back panel of the chassis (see Figure 16).

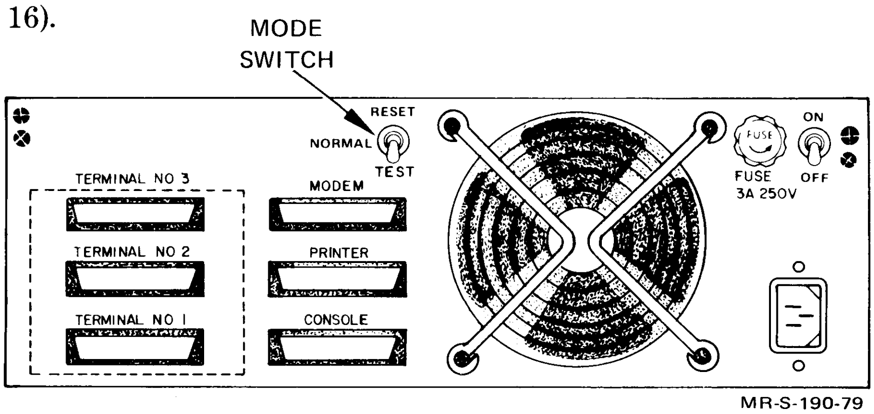


Figure 16. Back of the MiniMINC Chassis

The ports and their uses are:

MODEM	for sending and receiving data with the MINC NFT option via modems (devices used in long distance transmission of digital data), acoustic couplers, or direct cable
PRINTER	for sending data to a printing device (printer)
CONSOLE	for communication with the MiniMINC terminal
TERMINAL #3 TERMINAL #2 TERMINAL #1	for communication with most terminals, printers, and other serial ASCII equipment conforming to EIA standard RS-232C

The MODEM, PRINTER, and CONSOLE ports perform only their designated functions; the TERMINAL #3, TERMINAL #2, and TERMINAL #1 ports are identical, general-purpose ports that function with a variety of devices. Chapter 2 of the *MiniMINC Supplement* provides information and illustrations regarding the connection and use of serial ASCII equipment with MiniMINC.

SERIAL ASCII ROUTINES

Communication between MiniMINC and a device depends on programs that include a set of special MiniMINC routines. These routines (SET_SERIAL, CIN, and COUT) transfer characters between MiniMINC and a connected serial ASCII device and determine the speed at which the characters travel, how they are represented, and the method to be used for checking their integrity. Chapter 2 of the *MiniMINC Supplement* describes the routines in detail.

THE SERIAL ASCII DEMONSTRATION PROGRAM

Description of LISSJU

LISSJU, a serial ASCII program, creates Lissajous figures on a Tektronix 4662 interactive digital plotter.

When the program starts, you supply five values that determine what kind of Lissajous figure LISSJU will create. The program computes the graph coordinates for your Lissajous figure and then chains to PLOT, a program that controls the plotter. PLOT causes the plotter to draw a border, the labels, and then the Lissajous figure.

Lissajous figures are frequency patterns commonly generated on an oscilloscope by connecting a source of known frequencies to one set of deflection plates and an unknown frequency to the other set. A recognizable pattern results when one frequency is a harmonic or integral multiple of the other, thus revealing the unknown frequency.

The following line in LISSJU contains the parametric equations used in computing the graph coordinates (REM is the REMARK statement. It distinguishes a comment from the executable portion of a program. See *Book 3: MINC Programming Reference* for a description):

```
X=SIN(X1*M*I+X2)\Y=SIN(Y1*M*I+Y2)\REM THE LISSAJOUS EQUATIONS
```

where:

X1	is the horizontal multiplier
Y1	is the vertical multiplier
X2	is the horizontal phase in radians
Y2	is the vertical phase in radians
M	is 2π divided by one less than the number of points N
I	is an integer that progresses from 1 to N

The ratio of X1 to Y1 determines the actual shape of the Lissajous figure. These multipliers are the frequencies of the X and Y sine waves that combine together to form the Lissajous figure.

The phase shift resulting from the combined effects of X2 and Y2 causes the figure to "rotate" so that if you created a series of Lissajous drawings with identical multipliers (X1 and Y1), each drawing would be a different view of the same figure. Figures 17 through 19 show the rotation of a Lissajous figure with a 1/3 frequency ratio.

PLOT's Serial ASCII Routines

The SET_SERIAL routine in the PLOT program sets the attributes of the serial line connecting MiniMINC with the plotter. This routine adjusts the serial line each time you run the programs to create a Lissajous figure. Chapter 2 of the *MiniMINC Supplement* explains the SET_SERIAL routine in detail.

The COUT serial ASCII routines in PLOT signal the plotter when to print text and when to plot points. They then direct the

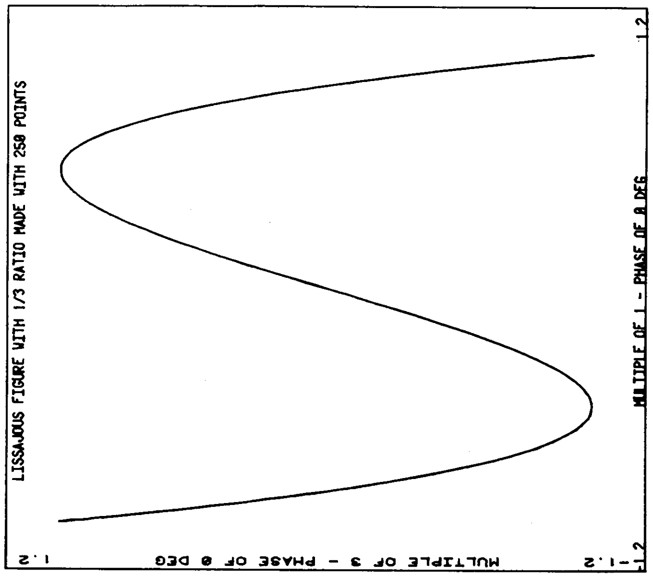


Figure 17.
Lissajous Figure
0°/0° Phase Shift

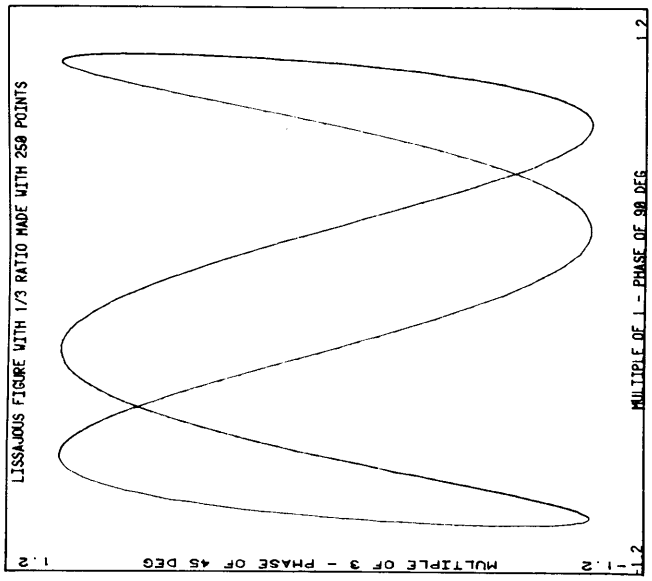
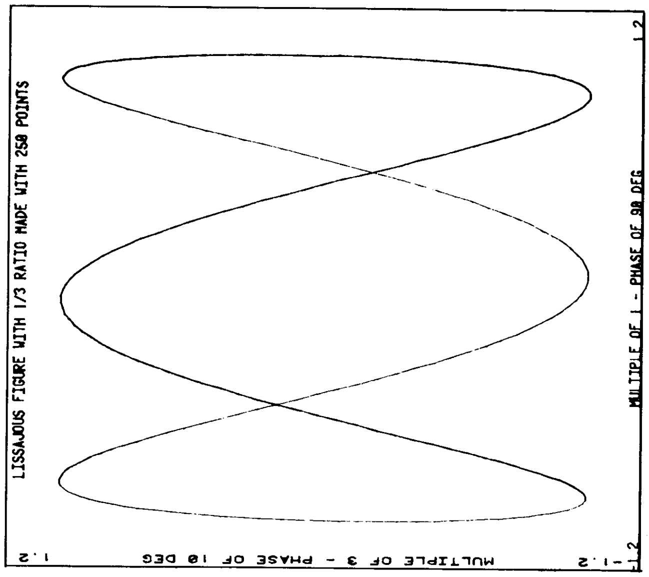


Figure 18.
Lissajous Figure
90°/45° Phase Shift



MR-S-192-79

Figure 19.
Lissajous Figure
90°/10° Phase Shift

plotter's pen to the proper coordinates, creating the final figure. Chapter 2 of the *MiniMINC Supplement* also describes this routine and its use.

The Procedure

Power up MiniMINC and the plotter (you can find the plotter's switch settings documented inside the PLOT and LISSJU programs). Next, insert the demonstration diskette into MiniMINC's drive unit 0 and engage the system. Then type:

```
RUN LISSJU (RET)
```

LISSJU responds:

```
LISSJU creates Lissajous figures on an interactive digital
plotter (see Chap. 4, Introduction to MiniMINC).

Note that LISSJU produces a figure of almost any
frequency ratio and phase shift. However, to produce
high-quality drawings, please confine your responses to the
following ranges:

1.  Ratio between horz/vert multipliers not to exceed 40:1
2.  Phases of 0 to 360
3.  No. of points no greater than 295
HORIZONTAL MULTIPLIER?
```

For the purpose of this demonstration, type the following values:

```
HORIZONTAL MULTIPLIER? 5 (RET)

VERTICAL MULTIPLIER? 6 (RET)

HORIZONTAL PHASE ANGLE IN DEGREES? 18 (RET)

VERTICAL PHASE ANGLE IN DEGREES? 18 (RET)

NO. OF POINTS TO USE (0-295)? 250 (RET)
```

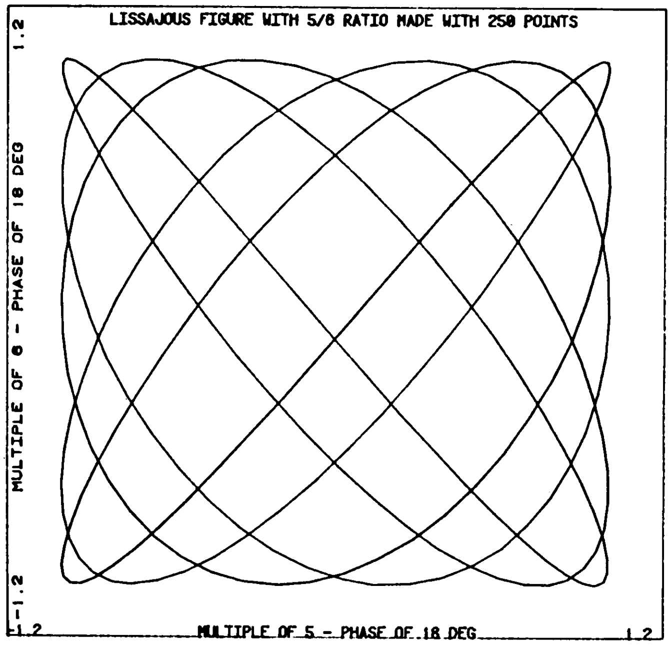
LISSJU then verifies your answers with the following messages:

```
LISSAJOUS FIGURE WITH 5/6 RATIO MADE WITH 250 POINTS

MULTIPLE OF 5 - PHASE OF 18 DEG

MULTIPLE OF 6 - PHASE OF 18 DEG
```

Approximately 50 to 60 seconds will pass before the plotter begins to draw. If you entered the values recommended for this example, the plotter will produce a figure like the one in Figure 20.



MR-S-193-79

Figure 20. Lissajous Figure — Plotter Sample

You can examine the LISSJU and PLOT programs with the OLD and LIST commands. Type:

```
OLD LISSJU (RET) or OLD PLOT (RET)
```

When MiniMINC responds with READY, type:

```
LIST (RET)
```

If you have a printer connected to the MiniMINC, type:

```
COPY LISSJU LP: (RET) or COPY PLOT LP: (RET)
```

CHAPTER 5
A MiniMINC PROGRAMMING PRIMER

This chapter describes how programs are used in MiniMINC and suggests a method you can use for developing programs. *Book 2: MINC Programming Fundamentals* discusses the MINC BASIC language and how to program in it.

PROGRAMS — A DESCRIPTION

A program is a set of computer instructions or symbolic statements combined to perform some task.

MiniMINC has two types of programs: system programs that make up the programming system and user programs (also known as application programs) that perform the special tasks required by the user. There are programs of both types supplied with MiniMINC — system programs and some user programs that demonstrate MiniMINC's capabilities.

MiniMINC system programs are machine language programs that are immediately intelligible to the computer without the intervention of a translating facility (the high-level MINC BASIC language requires interpretation before execution). System programs control and direct the computer's activities.

MiniMINC user programs are series of MINC BASIC statements combined to perform tasks required by users. Such tasks include data processing, graphic displays, and serial ASCII data transfers.

CREATING USER PROGRAMS

Writing a program to run on MiniMINC requires a programming method and knowledge of the MINC BASIC language and program structure. *Book 2: MINC Programming Fundamentals* explains MINC BASIC and this chapter explains a programming method.

There are probably as many ways to write a program as there are programmers, but not every method results in an efficient and effective program. This chapter describes one method that works. Note that the following pages describe steps to take before finally writing the program in the MINC BASIC language.

Defining a Program's Objective

First determine what you want the program to do.

Suppose you want a program that arranges names in a list. There are some basic questions you must answer before you can write the program. For instance:

- How should the names be ordered (alphabetically, randomly, order of importance)?
- In what form should the names appear (full name, last name only, last name and first initial)?
- How many names will the list contain (fixed or variable number)?
- How will the computer receive the names (typed at the terminal or from a file on diskette)?
- In what form should the results appear (on the terminal screen, in a file, or on paper from a printer)?

The answers to questions like these help to clarify the program's objective.

A definition of the program's objective can help you to judge the program's feasibility in terms of available time and resources. For instance, your potential program may require a printer when your system lacks one or you may not have the time necessary to type long lists of names should they be required. Whatever the limiting factor or factors (computer memory size, limited time, unavailable equipment, or insufficient data), you can adjust your program's objective accordingly. Thus a comprehensive definition at the outset of your project can save you time and frustration later on while you're programming.

Pretend, for the sake of illustration, that you need a program to sort a list of names. Your requirements are for a program that will:

- accept up to 1000 names from the terminal
- use names in the form of last name and first initial
- sort names alphabetically
- display the names in correct order on the terminal screen

This program already exists as the demonstration program ALPHA. The remainder of this chapter traces ALPHA's development up to the final programming effort.

Outlining a Solution

The next step in building a program is to establish your programming strategy.

Create an outline for your program by describing the program's method of operation in general terms. Your outline should include every process required by your program to achieve its objective. The processes should account for the input of data, data processing, and the output of data. When creating the outline, list the processes according to their sequence in the program.

For example, you can outline ALPHA in the following way:

1. Input names
2. Reorder names
3. Output reordered names

This outline represents everything the program must do. Step 1 is data entry, Step 2 is data processing, and Step 3 is the result.

Outlining a solution increases the likelihood that you will develop an effective program. If you are satisfied that the outline contains all the processes required by the program's objective, then proceed to the next step. The ability to define a process will develop as you learn the MINC BASIC program structure described in *Book 2: MINC Programming Fundamentals*.

Constructing an Algorithm

An *algorithm* is a fixed series of well-defined procedures that solves a problem in a finite amount of time. It is the essence of a program.

An algorithm summary is the written representation of an algorithm. The summary often takes the form of brief English sentences or statements that succinctly describe each procedure within the algorithm. These sentences or statements appear in the summary in the same order as the procedures they describe.

Consider the outline for ALPHA. The first process, inputting the names, uses an algorithm described in the following summary:

1. Create array for storing unordered names
2. Receive name from terminal
3. If no more names, then go to Step 6
4. Store name in unordered array
5. Go to Step 2
6. Stop

The procedures are performed in numerical order. However, some procedures override this order by directing control to a specific procedure as in steps 3 and 5. Step 3 switches control to Step 6 when you are finished typing names. The switching procedure in Step 3 reflects a MINC BASIC programming technique called a *conditional branch.* Step 5 always passes control to Step 2. This switching procedure reflects an *unconditional branch.* You should always take care to limit the number of branches of either kind. Algorithm summaries (and programs) can become confusing unless most of the activity is performed in simple numerical order.

You may wonder why there are so many steps in this algorithm summary. This is because each procedure corresponds to a MINC BASIC statement or set of statements that you will learn about in Book 2.

For the moment, concentrate on the form of the algorithm summary. Mentally test the algorithm. The more you remember of its pattern, the better you'll understand the discussions on program structure in Book 2. This is true because a program is an algorithm translated into action by the use of a programming language.

The following algorithm summaries show how ALPHA reorders and then outputs names.

REORDER NAMES

1. Create array to store ordered names
2. Take next name from unordered array
3. Compare name with next name in unordered array
4. Take name with lowest alphabetical position, switching position of names if necessary
5. If name compared with last element of unordered array, then go to Step 7
6. Go to Step 3
7. Insert name in first open position of ordered array
8. If every name in unordered array tested for alphabetical order, then go to Step 10
9. Go to Step 2
10. Stop

OUTPUT ORDERED NAMES

1. Display next name in ordered array
2. If end of ordered array, then go to Step 4
3. Go to Step 1
4. Stop

NOTE

The word "next" in the algorithm summaries can also mean "first." This practice reflects how the BASIC language handles arrays.

Examine these algorithm summaries carefully. Notice how explicit they are. This exactness is required because programs direct the computer's operation, and computers perform only as directed. If you write a program based on an ambiguous algorithm, you must compensate for the ambiguity while programming. However, because programming involves much more detail than constructing algorithms, allowing ambiguity into your algorithm summary greatly increases your work load.

The three algorithm summaries given in this chapter should be combined into one summary to represent the entire ALPHA program. The resulting algorithm summary for ALPHA is:

1. Create array for storing unordered names
2. Receive name from terminal
3. If no more names, then go to Step 6
4. Store name in unordered array
5. Go to Step 2
6. Create array to store ordered names
7. Take next name from unordered array
8. Compare name with next name in unordered array
9. Take name with lowest alphabetical position, switching position of names if necessary
10. If name compared with last element of unordered array, then go to Step 12
11. Go to Step 8
12. Insert name in first open position of ordered array
13. If every name in unordered array tested for alphabetical order, then go to Step 15
14. Go to Step 7
15. Display next name in ordered array
16. If end of ordered array, then go to Step 18
17. Go to Step 15
18. Stop

Notice that the "stops" formerly at the end of each process algorithm are missing. They were needed because a *loop* (a repeating sequence of events) inside each process terminated itself and the process by sending control to a stop. Now that the processes are combined, control passes to the next process instead of terminating at a stop. The stop at Step 18 remains to terminate the ALPHA algorithm.

The algorithm described for ALPHA is only one of several algorithms capable of the same function. In fact, there is almost always more than one solution to any given programming problem. Keeping this in mind, you should evaluate your program objectives carefully in order to construct the most efficient and effective algorithms possible. A useful technique that can improve your understanding of a process, as well as the quality of the final algorithm, is to develop several algorithms for the same program.

Testing an Algorithm

After constructing an algorithm, and in its early stages of development, you should test the algorithm for integrity of design or soundness.

To be sound, an algorithm must solve a problem in a finite amount of time with a fixed series of well-defined processes. Every process required to solve the problem must therefore be included within the algorithm and occur in proper sequence. There must also be a point in time when all the processes are finished and the problem is solved.

A method for appraising an algorithm's design integrity is to mentally trace the path of several data items through the algorithm, one item at a time. You can detect flaws with this method that you would probably overlook in a general inspection.

Try this method on ALPHA's algorithm. Perform each step in the algorithm separately and in the indicated order. For now, and whenever you test an algorithm in this manner, suspend your intuition and perform each step exactly as described.

Using a piece of paper as the unordered array, write down a name each time you encounter Step 4, until you've entered four or five names (any names will do). When you've entered all the names, Step 3 directs you to Step 6. Continue with the algorithm until you reach the stop in Step 18. Use additional paper to represent the ordered array and ALPHA's output.

This simple test method should work for all programming algorithms. Proceed to the next stage in program development, implementing the algorithm, when you are convinced of the algorithm's soundness.

Implementing an Algorithm

An implemented algorithm is a program. The writing of a program requires knowledge of a programming language and program structure.

You should next read *Book 2: MINC Programming Fundamentals*. This book explains what you need to know to begin programming in MINC BASIC. The other MiniMINC manuals (*Book 3: MINC Programming Reference, Book 4: MINC Graphic Programming*, and the *MiniMINC Supplement*) describe the MINC BASIC commands, statements, and routines.

SUMMARY

The MiniMINC system uses two types of programs: system and user programs. System programs control and direct the computer's operations and thus form part of the MiniMINC system. User programs are created by MiniMINC users to perform tasks with the system. The system programs are machine language programs and the user programs are high-level language programs.

There are many ways to create user programs. This manual demonstrates one of them. The steps involved are:

Define a program's objective

Outline a solution

Construct an algorithm

Test the algorithm

Implement the algorithm

Book 2: MINC Programming Fundamentals explains how to implement an algorithm with the MINC BASIC language.

CHAPTER 6
DIRECTIONS FOR READING THE MiniMINC MANUALS

EXPERIENCED PROGRAMMERS

If you are an experienced programmer in a language other than BASIC, we recommend that you follow the reading order suggested below for novice users.

If you are familiar with BASIC, but not MINC BASIC, you should at least skim Books 2 and 3 for new commands and the MiniMINC approach to standard commands.

NOVICES

Book 2: MINC Programming Fundamentals continues your education in MiniMINC programming, but concentrates on more practical matters such as file creation and editing. It teaches MINC BASIC fundamentals in a thorough, uncomplicated manner using examples. You should find Book 2 and Chapter 5 of this manual sufficient preparation for programming on your own.

You will probably need to review sections of Book 2 before you fully understand MINC BASIC as a "language." Review and practice are the keys to a functional understanding of MINC BASIC or any other programming language. Only by gaining experience will you achieve programming proficiency.

You should find *Book 3: MINC Programming Reference* useful once you start programming. Book 3 and the *MiniMINC Supplement* together describe every command, statement, function, and routine in MINC BASIC.

Book 4: MINC Graphic Programming teaches the concepts underlying graphics and conducts you through some demonstrations. Its second half serves as a reference manual for graphic routines once you begin graphic programming. We recommend that you read it thoroughly beforehand.

Reserve the *MiniMINC Supplement* for last. This book contains technical information for connecting MiniMINC to printers and other devices, testing your equipment, and correcting problems that may arise. The Supplement also describes new features that Books 1 through 4 don't point out.

APPENDIX A HANDLING UNUSUAL CONDITIONS

This section helps you identify and correct minor problems that might occur when you use the system. These problems usually result from loose connections or user oversights that you will learn to avoid with experience.

Condition	Procedure
You type two @ signs but the system doesn't identify itself or request the date.	Check lamp 2 on the chassis. If it's on, the chassis is waiting for another @ sign or two.
	Ensure that a demonstration or system diskette is in drive unit 0 and that the drive unit's door is closed properly.
	Check that the power switches for the terminal and chassis are in the up position.
	Check that the ON LINE light is on. This light is part of the horizontal array of lights located at the top of the terminal's keyboard (near the SET-UP key). If the LOCAL light glows instead, press the SET-UP key, then the "4" key below the lights, and then the SET-UP key once again. The

Condition	**Procedure**
	ON LINE light must always be on for you to communicate with the system. Check that the power cords are plugged into the proper outlets, that the cable connecting the terminal and the chassis is secure at both ends, and that the chassis' POWER lamp is lit.

You get one of the following error messages when you type two @ signs to start the system:
 	
`BOOT-F-I/O error`
 	
`MON-F-System read failure halt`
	Check that a demonstration or system diskette is in drive unit 0 and that the door is properly closed. Try again.

You are using MiniMINC and an @ character with trailing digits appears on the terminal screen.	Hold down the SHIFT key and press the "P" key. If MiniMINC doesn't respond, turn off the chassis and restart the system according to the directions in Chapter 2, Powering the System. Data not stored on diskette are lost.
You type characters on the keyboard but they don't appear on the terminal screen.	Press the NO SCROLL key and try typing some more characters. Continue working if the characters appear on the screen. If characters don't appear, press the NO SCROLL key to reenable scrolling and then check that the keyboard is plugged securely into the terminal.

Condition	Procedure
	If characters don't appear, hold down the CTRL key and press the "Q" key.
	If characters don't appear, press the SET-UP key, then the RESET(0) key.
	If MiniMINC still doesn't respond, turn off the chassis and restart the system according to Chapter 2, Powering the System.
	Follow the earlier recovery procedures in this appendix if the trouble persists.
The words "SET UP" appear in the upper left corner of the terminal screen.	Press the SET-UP key. The screen should return to normal.
	If the trouble persists, turn the terminal off and then on again.
	If this doesn't help, turn off the chassis and then restart the system following the instructions in Chapter 2, Powering the System.
	If you experience any further difficulty, consult Chapter 2 in the *MiniMINC Supplement*, Changing Operating Modes on the MiniMINC Terminal.

If any problem persists after you've tried these recovery procedures, refer to the Troubleshooting section of Chapter 2 in the *MiniMINC Supplement*.

APPENDIX B
ERROR MESSAGE COMPENDIUM

This appendix lists the error messages produced by MiniMINC. The Messages section of *Book 3: MINC Programming Reference* contains additional information on error messages.

HOW TO USE THE ERROR MESSAGE COMPENDIUM

When an error message appears on the MiniMINC terminal screen, you can either correct the problem immediately (some MiniMINC error messages explain what caused the error and tell you what to do about it) or you can look up the message in this compendium.

Error messages are represented in this compendium almost exactly as they appear on the screen. The form of an error message is:

?ENVIRONMENT-S-Text

where:

?	precedes all error messages (the error message compendium disregards this symbol because it is common to all MiniMINC error messages).
ENVIRONMENT	identifies the part of the MiniMINC programming system (environment) that produced the error. The environments are BOOT, DIR,

	EDITOR, FORMAT, HELP, KMON, MINC, MON, and UTILITY.
S	indicates the error's severity; F for fatal, W for warning, or I for information. A fatal error message means that MiniMINC halted execution of your command, statement, or program. A warning error message means that MiniMINC didn't halt execution of your command, statement, or program. However, you should correct the problem and then try again. An informational error message gives information about the environment without halting the program.
Text	explains the problem and sometimes tells how to fix it. Some error messages identify what line in your program caused the error.

Error messages are alphabetically arranged by environment, severity, and text respectively.

Some of the entries contain references to other MiniMINC books. For example,

MINC-F-Argument outside terminal scrolling limits
Book 4, see: ROLL_AREA

means that Book 4 explains the error message in its description of the ROLL_AREA graphics routine.

Other entries give an explanation of the error and a recovery procedure. For example,

MINC-F-No suitable free space on volume for file
Directory may be full; try COLLECT. If necessary, delete some files and try COLLECT again. Otherwise, try another volume.

The error messages listed in this book result from errors that you may encounter under normal operating conditions. If you receive an unlisted error message, try the following procedure:

1. Check that the diskette drives are properly loaded and that the diskette drive doors are closed.

2. Repeat the instructions you gave MiniMINC just before the error message appeared.

3. If the error message returns (or the message tells you to notify DIGITAL), contact the MINC Product Services Center between the hours of 9 A.M. and 6 P.M., EST. To reach the MINC Product Services Center, eligible customers should call the number corresponding to their location.

Massachusetts	1-800-762-9700
The continental United States (excluding Massachusetts)	1-800-225-9366
Hawaii, Alaska, and Canada (call collect)	0-617-493-9473

BOOT ERROR MESSAGES

BOOT-F-I/O error
System diskette probably bad; make new copy.

BOOT-F-No boot on volume
Book 3, see: Starting Procedures
Book 2, see: MINC Volumes

DIR ERROR MESSAGES

DIR-F-Error reading directory
Book 3, see: DIRECTORY

DIR-F-Illegal device
Book 3, see: File Specifications

DIR-F-Illegal directory
Book 3, see: DIRECTORY

DIR-F-Output file full
Book 3, see: DIRECTORY

DIR-F-Write error
Device detected an error in writing; check diskette drive and try again. Try another diskette if the message reappears.

EDITOR ERROR MESSAGES

EDITOR-F-Cannot find input file on specified or default volume
Book 3, see: EDIT

EDITOR-F-Cannot proceed; owner is 'DIGITAL' for output volume
Diskettes owned by DIGITAL do not accept alterations. Either replace the diskette or direct your output file to the other diskette drive.

EDITOR-F-No output volume space large enough to EDI input file
Book 3, see: EDIT

EDITOR-F-Non directory device illegal for output
Book 3, see: File Specifications

EDITOR-F-Output volume has maximum number of files or no free blocks
Book 3, see: CREATE, EDIT

EDITOR-F-Unable to check volume owner
Detected an error in reading; check diskette drive and try again. Try another diskette if the message reappears.

EDITOR-F-Unable to size screen
Book 3, see: EDIT

EDITOR-W-Abort edit session losing all edits (Y,N)?
Book 2, see: Control Characters and the Keypad Editor

EDITOR-W-I/O error; check diskette drives and message manuals
The file contains a bad block; see Book 3 for a discussion of bad blocks.

EDITOR-W-Limited space for insertions (only # blocks). Continue (Y or N)?
Book 3, see: CREATE, EDIT

EDITOR-W-Output file name is already in use.
Do you want to erase its current contents (Y or N)?
Book 2, see: The Three Editing Commands

FORMAT ERROR MESSAGE

FORMAT-F-Device error
Indicates serious disk corruption; try a new diskette. If the problem persists, notify the MINC Product Services Center.

HELP ERROR MESSAGES

HELP-F-Help not available for #
Book 3, see: HELP

HELP-F-Syntax error in command, type HELP <ret>
Book 3, see: HELP

HELP-W-Default system HELP file not found
No help for topic #
Book 1, see: Getting HELP

HELP-W-No help for topic HELP #
Help topic is invalid; type HELP to determine valid topics.

KMON ERROR MESSAGES

KMON-F-Bad Fetch
Indicates serious disk corruption; try a new diskette. If the problem persists, notify the MINC Product Services Center.

KMON-F-Command file I/O error
Indicates serious disk corruption; try a new diskette. If the problem persists, notify the MINC Product Services Center.

KMON-F-Command file not found
Indicates serious disk corruption; try a new diskette. If the problem persists, notify the MINC Product Services Center.

KMON-F-File not found
Indicates serious disk corruption; try a new diskette. If the problem persists, notify the MINC Product Services Center.

KMON-F-Handler file I/O error
Indicates serious disk corruption; try a new diskette. If the problem persists, notify the MINC Product Services Center.

KMON-F-Overlay read error
Indicates serious disk corruption; try a new diskette. If the problem persists, notify the MINC Product Services Center.

KMON-W-Illegal date
Book 3, see: DATE

KMON-W-Illegal time
Book 3, see: TIME

MINC ERROR MESSAGES

MINC-F-An earlier statement already defined the function
Book 2, see: User-Defined Functions
Book 3, see: DEF

MINC-F-Another virtual array is using the channel
Book 3, see: DIM

MINC-F-Arguments in definition do not match function called
Book 3, see: ASC, ATN, BIN, CHR$, COS, INT, LEN, OCT, POS, SEG$, SGN, SQR, STR$, TAB, TRM$, TTYSET, VAL

MINC-F-Argument outside terminal scrolling limits
Book 4, see: ROLL_AREA

MINC-F-Array has invalid description
Book 3, see: COMMON, DIM

MINC-F-Array length is too small
Book 4, see: DUAL_MOVE, GRAPH

MINC-F-Array overfills workspace
Book 3, see: Arrays, DIM, LENGTH

MINC-F-Array subscript negative or too large
Book 2, see: Nested Loops, Subscripts
Book 3, see: Arrays

MINC-F-Bad column specification
Book 4, see: BOX, GET_CURSOR, MOVE_CURSOR, HTEXT, PUT_SYMBOL, TEXT_LINE, VTEXT

MINC-F-Bad row specification
Book 4, see: BOX, GET_CURSOR, MOVE_CURSOR, HTEXT, PUT_SYMBOL

MINC-F-Both REAL and IMAG must be at least as large as SIZE
MiniMINC Supplement, see: FFT, POWER

MINC-F-CALL fails; workspace too full for parameters
Book 3, see: Routines

MINC-F-Cannot erase a currently unused graph
Book 4, see: ERASE_GRAPH

MINC-F-Cannot find routine named
Book 3, see: CALL

MINC-F-Cannot get cursor position from terminal
Book 4, see: GET_CURSOR

MINC-F-Cannot specify X value less than 1 with DISTANCE
Book 4, see: DUAL_MOVE

MINC-F-Can't be in double strip-chart mode when using this routine
Book 4, see: POINT

MINC-F-Can't call REGION when in strip-chart (move) mode
Book 4, see: REGION

MINC-F-Can't display vertical lines in strip-chart (move) mode
Book 4, see: GRAPH

MINC-F-Can't turn on graph that's not in use
Book 4, see: VIEW

MINC-F-Channel or unit # not in system for the routine
MiniMINC Supplement, see: CIN, COUT, FFT, SCHEDULE

MINC-F-Check the manual; DUP cannot accept any file name
Book 3, see: DUPLICATE

MINC-F-COMMON variables not in same order as in last program
Book 2, see: Preserving Values of Variables in a Chain
Book 3, see: CHAIN,COMMON

MINC-F-COP requires 2 file names
Book 3, see: COPY

MINC-F-Could not find service subroutine # requested
MiniMINC Supplement, see: SCHEDULE

MINC-F-Data lost—transfer rate too high
MiniMINC Supplement, see: CIN

MINC-F-Data transfer or pending service request terminated
MiniMINC Supplement, see: PAUSE,SCHEDULE

MINC-F-DATA value or value from file does not match variable
Book 2, see: READ and DATA Statements
Book 3, see: DATA, READ

MINC-F-Distance must be specified if X is omitted
Book 4, see: DUAL_MOVE

MINC-F-Earlier array description differs from reference
Book 3, see: DIM

MINC-F-END statement does not have highest number in program
Book 3, see: END

MINC-F-EOF error in compiled program; use program stored with SAVE
The file produced by the COMPILE command contains a format error; use a copy of the program created by a SAVE or REPLACE command.

MINC-F-Error in compiled program; use program stored with SAVE
The file produced by the COMPILE command contains a for-

mat error; use a copy of the program created by a SAVE or REPLACE command.

MINC-F-File channel input or output error
Accessing data in a file produced an error; check diskette and diskette drive.

MINC-F-File channel is not in the range 1-12
Book 3, see: CLOSE

MINC-F-File name in use; REPLACE or change name or volume
Book 2, see: Program Files, The REPLACE Command
Book 3, see: REPLACE, SAVE

MINC-F-File space allocated on volume is too small
Book 3, see: COMPILE, OPEN, PRINT, PRINT USING

MINC-F-First argument must be smaller than second
Book 4, see: ROLL_AREA

MINC-F-GOSUB fails; 20 routines already active
Book 3, see: GOSUB, ON, RETURN

MINC-F-Graph number must be zero when DUAL_MOVE used
Book 4, see: ERASE_GRAPH

MINC-F-Graph number of zero not allowed here
Book 4, see: BARGRAPH, GRAPH, GRID, HLINE, MAP_TO_TEXT, POINT, VLINE

MINC-F-Graphics routine terminated by ^C
You typed the CTRL/C signal to interrupt a graphics routine.

MINC-F-Illegal baud rate specified
MiniMINC Supplement, see: SET_SERIAL

MINC-F-Illegal graph number
Book 4, see: GRAPH, GRID, ERASE_GRAPH, HLINE, LABEL, MAP_TO_TEXT, POINT, SHADE, VIEW, WINDOW, REGION, BARGRAPH

MINC-F-Illegal number of bits specified
MiniMINC Supplement, see: SET_SERIAL

MINC-F-Incomplete command; COP requires 2 file names; TYP requires at least one
Book 3, see: COPY

MINC-F-Index array is not present
Book 4, see: GRAPH

MINC-F-Indexing not allowed in strip-chart (move) mode
Book 4, see: GRAPH

MINC-F-Invalid command; use only 2 file names
Book 3, see: COPY, DIR, INI, TYPE

MINC-F-Invalid data type for argument #
There was an invalid data type given for an argument in the position listed.

MINC-F-Invalid file name(s)
Book 3, see: APPEND, DIRECTORY, NEW, RENAME, SAVE

MINC-F-Invalid operation; mixing numbers and strings
Book 3, see: Assignment statement, DEF

MINC-F-Invalid or conflicting options requested
Book 4, see: BOX, CHAR_MODE, DISPLAY_MODE, DUAL_MOVE, ERASE_GRAPH, ERASE_TEXT, GRAPH, BARGRAPH, WIDE_LINE, HTEXT, VTEXT, LIGHTS, POINT, PUT_SYMBOL, REGION, SHADE, TEXT_LINE
MiniMINC Supplement, see: CIN, COUT, SCHEDULE

MINC-F-Invalid PRINT USING format or syntax
Book 2, see: PRINT USING Statement Error Conditions
Book 3, see: PRINT USING

MINC-F-I/O error; unable to check volume owner
Book 3, see: NAME, SAVE

MINC-F-Line is longer than 132 characters
The line is ignored; break the line into two or more lines. Or, if reading from a file, make sure the file contains only legal BASIC program lines.

MINC-F-Line is too long to translate
The line is ignored; break the line into two or more lines. Or, if reading from a file, make sure the file contains only legal BASIC program lines.

MINC-F-Line must have different endpoints
Book 4, see: TEXT_LINE

MINC-F-Missing argument no. # is required
Omitted required argument in the position listed.

MINC-F-MOVE and BRAND cannot be specified together
Book 4, see: GRAPH, POINT

MINC-F-MOVE option must be selected if X argument is omitted
Book 4, see: POINT

MINC-F-MOVE option must be specified if Move specified previously
Book 4, see: GRAPH, POINT

MINC-F-MOVE option must be specified to specify Units
Book 4, see: POINT

MINC-F-MOVE option not allowed if other graph is in use
Book 4, see: GRAPH, POINT

MINC-F-Need OPEN statement for file channel
Book 3, see: CLOSE, INPUT, LINPUT, OPEN, PRINT, PRINT USING, RESTORE/RESET

MINC-F-No corresponding FOR statement for NEXT
Book 2, see: Nested FOR Loops
Book 3, see: FOR, NEXT

MINC-F-No DEF statement for function named
Book 2, see: User-Defined Functions
Book 3, see: DEF

MINC-F-No DIM or COMMON description for array used
Book 3, see: Arrays

MINC-F-No NEXT statement terminates FOR loop
Book 3, see: FOR, NEXT

MINC-F-No suitable free space on volume for file
Directory may be full; try COLLECT. If necessary, delete some files and try COLLECT again. Otherwise, try a different volume.

MINC-F-No workspace available for the string specified
MiniMINC Supplement, see: CIN

MINC-F-Notify DIGITAL: Internal error trap
This message does not appear when MiniMINC is functioning properly. It indicates a serious problem in the MiniMINC routines.

MINC-F-Notify DIGITAL: Mark time failure
This message does not appear when MiniMINC is function-

ing properly. It indicates a serious problem in the MiniMINC routines.

MINC-F-Notify DIGITAL: Memory pool exhausted
This message does not appear when MiniMINC is functioning properly. It indicates a serious problem in the MiniMINC routines.

MINC-F-Notify DIGITAL: Protection failure
This message does not appear when MiniMINC is functioning properly. It indicates a serious problem in the MiniMINC routines.

MINC-F-Number of indices is outside range 0-10
Book 4, see: GRAPH

MINC-F-Number of indices is too large for specified array
Book 4, see: GRAPH

MINC-F-Only one graph number may be in use
Book 4, see: FIND_POINT

MINC-F-Only one parity option may be specified
MiniMINC Supplement, see: SET_SERIAL

MINC-F-Only one symbol may be printed at a time!
Book 4, see: PUT_SYMBOL

MINC-F-OPEN fails; file channel already open
Book 3, see: OPEN

MINC-F-OPEN fails; workspace too full for another channel
Book 3, see: OPEN

MINC-F-OPEN statement for file channel prohibits transfer
Book 3, see: OPEN, RESTORE/RESET

MINC-F-Outer FOR loop is using control variable
Book 3, see: FOR, NEXT

MINC-F-Output file name already in use;
do you want to erase its current contents (Y or N)?
Book 3, see: COPY

MINC-F-Parentheses nested too deeply or too many user functions
Break the statement into several simpler ones.

MINC-F-Previously plotted point not found
Book 4, see: GRAPH

MINC-F-Previously scheduled event pending. No new request
MiniMINC Supplement, see: SCHEDULE

MINC-F-Program does not have statement number specified
Book 3, see: CHAIN, GOSUB, GO TO, ON

MINC-F-Program too large; workspace overfills
Book 3, see: APPEND, CHAIN, COMMON

MINC-F-Quotes around string do not match or end quote missing
Book 2, see: String Literals

MINC-F-Range of WINDOW limits too narrow
Book 4, see: WINDOW (Restrictions)

MINC-F-RESEQ has an invalid statement number or interval
Book 3, see: RESEQ

MINC-F-Reached RETURN without executing a GOSUB statement
Book 3, see: GOSUB, ON, RETURN

MINC-F-Serial line number must be in range 1-3
MiniMINC Supplement, see: SET_SERIAL

MINC-F-Serial output channel # is not ready
MiniMINC Supplement, see: COUT

MINC-F-Single strip-chart mode cannot be on when DUAL_MOVE is called
Book 4, see: DUAL_MOVE

MINC-F-SIZE is greater than 2048
MiniMINC Supplement, see: FFT

MINC-F-SIZE is less than or equal to 8
MiniMINC Supplement, see: FFT

MINC-F-SIZE is not a power of 2
MiniMINC Supplement, see: FFT

MINC-F-SIZE must be greater than or equal to REAL, IMAG, & PSPECT arrays
MiniMINC Supplement, see: POWER

MINC-F-SIZE must be positive
MiniMINC Supplement, see: POWER

MINC-F-Sorry; the statement is invalid in immediate mode
Not all statements are valid in immediate mode; include the statement in a short program and run the program.

MINC-F-Specified corners are not diagonally opposite
Book 4, see: BOX

MINC-F-Specified number not convertable to text coordinate
Book 4, see: MAP_TO_TEXT

MINC-F-Specified options incompatible with moving graphs
Book 4, see: GRAPH, ERASE_GRAPH

MINC-F-Specified or default volume does not have file named
Book 2, see: Running Programs
Book 3, see: APPEND, CHAIN, KILL, NAME, OPEN, REPLACE, RUN/RUNNH, UNSAVE

MINC-F-Specified Row or Column not in graphic region
Book 4, see: MAP_TO_GRAPH

MINC-F-Specify device only; no filename or file type is allowed
Book 3, see: VERIFY

MINC-F-Specify either LP: or serial line number, but not both
MiniMINC Supplement, see: SET_SERIAL

MINC-F-Specify only one file name
Book 3, see: VERIFY

MINC-F-Specify only one region, please
Book 4, see: REGION

MINC-F-String is longer than 255 characters
Book 3, see: Assignment

MINC-F-String length must be from 0 to 255 characters
MiniMINC Supplement, see: CIN, COUT

MINC-F-Stripchart modes must be off when this routine is called
Book 4, see: FIND_POINT

MINC-F-SUB creates an invalid statement or has a syntax error
Book 2, see: The Substitute Command
Book 3, see: SUB

MINC-F-Syntax error; cannot translate the statement
Book 3, see: APPEND, Arithmetic, DIM, LINPUT, RUN/RUNNH

MINC-F-The TEXT option must be specified by itself
Book 4, see: ERASE_GRAPH

MINC-F-Time string format must be hh:mm:ss
MiniMINC Supplement, see: PAUSE, SCHEDULE

MINC-F-Too few values for INPUT or READ variables
Book 2, see: Checking for the End of the Input File
Book 3, see: DATA, INPUT, LINPUT, READ

MINC-F-Too many arguments in statement
This message appears when there are more arguments in the argument list than the routine can interpret. This condition can arise because of extra commas in the argument list.

MINC-F-Too many response requests pending
MiniMINC can delay responding to response requests until previous requests are handled, but at most eight response requests can be pending at one time. SCHEDULE causes response requests.

MINC-F-Total number of arrays and variables in COMMON exceeds 255
Book 3, see: COMMON

MINC-F-Use another file type; SYS, SAV, COM, and BAD are protected
Book 3, see: Protected File Types, UNSAVE

MINC-F-Use array element instead of array for argument #
MiniMINC Supplement, see: CIN, COUT

MINC-F-Value of argument # is outside valid range
Book 4, see: GRAPH, DUAL_MOVE, ERASE_TEXT, HLINE, VLINE
MiniMINC Supplement, see: PAUSE, SCHEDULE

MINC-F-Value of control expression is out of range
Book 2, see: ON/GO TO Statements
Book 3, see: ON

MINC-F-Value of FILESIZE expression too large or less than -1
Book 3, see: OPEN

MINC-F-Variable name required for argument #
An argument was a literal instead of the required variable or array name. This condition can result from missing or extra commas in the argument list.
Book 4, see: FIND_POINT, GET_CURSOR, MAP_TO_GRAPH, MAP_TO_TEXT
MiniMINC Supplement, see: FIND_CURSOR, GET_CHAR

MINC-F-Width + Distance must be less than 512
Book 4, see: WIDE_LINE

MINC-F-Workspace contents remain unchanged
Not enough memory available to store all the strings used in the program; reduce the size of the BASIC program.
Book 3, see: EXTRA_SPACE, NORMAL_SPACE

MINC-F-Wrong number of arguments or improper argument default
There was an improper number of arguments for a MINC BASIC routine.

MINC-F-X value must be less than 512 when DISTANCE is selected
Book 4, see: DUAL_MOVE

MINC-F-Zero is illegal as a ROLL_AREA argument
Book 4, see: ROLL_AREA

MINC-W-Attempt to find square root of a negative value
Book 3, see: SQR

MINC-W-Cannot recover workspace program; see message manual
Indicates user workspace program and variables have been lost. Usually occurs when you request a directory after the EXTRA_SPACE command.

MINC-W-Dividing by zero
Book 2, see: Multiplication, Division
Book 3, see: Arithmetic

MINC-W-Enter new value. Old value did not match INPUT variable
Book 3, see: INPUT

MINC-W-Extra values from keyboard or file ignored
Book 3, see: INPUT

MINC-W-Invalid exponential expression
The program tried to compute the value A^B where A<0 and B was not an integer or B>256. BASIC substitutes a value or zero for the result.

MINC-W-LOG or LOG10 expression less than or equal to 0
Book 3, see: LOG, LOG10

MINC-W-Value of integer expression not in range -32768 to +32767
Book 3, see: Arithmetic, Assignment Statement

MINC-W-Value of real expression is too large
Book 3, see: Arithmetic, Assignment Statement, ATN, EXP

MINC-W-Value of real expression is too small
Book 3, see: Arithmetic, Assignment Statement, ATN, EXP

MON ERROR MESSAGES

MON-F-Bad fetch
Indicates serious disk corruption; try a new diskette. If the problem persists, notify the MINC Product Services Center.

MON-F-Directory I/O error
Book 3, see: COPY

MON-F-Directory overflow
Collect and try again; if condition persists, delete some files and collect again. Otherwise, try a different volume.

MON-F-Overlay error
Indicates serious disk corruption; try a new diskette. If the problem persists, notify the MINC Product Services Center.

MON-F-Power fail halt
Restart system.

MON-F-Swap error
Indicates serious disk corruption; try a new disk. If the problem persists, notify the MINC Product Services Center.

MON-F-System halt
Restart system.

MON-F-System read failure halt
Book 3, see: Starting Procedures

MON-F-Trap to 10
Possibly caused by static electricity; try again. If the condition persists, notify the MINC Product Services Center.

MON-F-Trap to 4
Possibly caused by static electricity; try again. If the condition persists, notify the MINC Product Services Center.

UTILITY ERROR MESSAGES

UTILITY-F-Bad directory block
Diskette not usable; try initializing diskette to recover it. If condition persists, discard diskette.

UTILITY-F-Copies of the following files may be faulty: #
Book 3, see: DUPLICATE

UTILITY-F-Device full
Book 3, see: DUPLICATE

UTILITY-F-Device SY may not be initialized
Book 3, see: INITIALIZE

UTILITY-F-Directory full
Collect and try again; if condition persists, delete some files and collect again or else use another diskette.

UTILITY-F-Disk unusable, too many bad blocks
Diskette not usable; try initializing diskette to recover it. If condition persists, discard diskette.

UTILITY-F-Error in formatting disk
Diskette not usable; try initializing diskette to recover it. If condition persists, discard diskette.

UTILITY-F-Error in system area
Diskette not usable; try initializing diskette to recover it. If condition persists, discard diskette.

UTILITY-F-Error reading directory
Book 3, see: COLLECT, VERIFY

UTILITY-F-Error writing directory
Book 3, see: INITIALIZE

UTILITY-F-File not found
Book 3, see: COPY, TYPE

UTILITY-F-Illegal command
Book 3, see: VERIFY

UTILITY-F-Illegal device
Book 3, see: COLLECT

UTILITY-F-Illegal directory
Book 3, see: DUPLICATE

UTILITY-F-Input truncated at twelve characters
Book 3, see: INITIALIZE

UTILITY-F-No swap file on boot volume
Put a valid MiniMINC diskette in drive 0 and restart the system.

UTILITY-F-No volume id
Book 3, see: DUPLICATE

UTILITY-F-Not enough usable space on volume
Book 3, see: COPY

UTILITY-F-Read error
Book 3, see: COLLECT, COPY, INITIALIZE

UTILITY-F-Target volume must be newly initialized
Book 2, see: Duplicating a Volume
Book 3, see: DUPLICATE

UTILITY-F-Unable to initialize volume with owner name 'DIGITAL'
Book 3, see: INITIALIZE

UTILITY-F-Unable to specify 'DIGITAL' as owner name
Book 3, see: INITIALIZE

UTILITY-F-Uninitialized volume
Book 3, see: COLLECT

UTILITY-F-Volume owner name may not be 'DIGITAL'
Book 3, see: COPY

UTILITY-F-Write error
Book 3, see: COPY

GLOSSARY

The glossary defines terms common to all of the MiniMINC manuals. A term is defined here if its use in the MiniMINC manuals deviates from its standard dictionary definition.

Algorithm
A fixed series of well-defined procedures that solves a problem in a finite amount of time. Programmers generally construct an algorithm as an outline for writing a new program.

Analog
Relating to the representation of a physical attribute such as voltage by an analogous phenomenon, such as the deflection in a graph.

Application
The particular task or set of related tasks for which you use a computer.

Application program
A program that performs a task specific to your needs. Usually, application programs are those programs not part of the system itself.

Argument
A variable or constant value or expression used with a command, statement, or routine that controls its action, specifically its location, direction, or range. An argument can also be any variable or constant value used by a function.

Array
An ordered arrangement of subscripted variables derived from a common variable. For instance, an array J might contain the subscripted variables J(0),J(1),J(2), and J(3).

ASCII
The American Standard Code for Information Interchange; a standard code for upper- and lower-case letters, numbers, punctuation, and special communication control characters.

Assembly language
A symbolic programming language that normally translates directly into a machine language.

Autoscaling
A MiniMINC feature that automatically adjusts the axis units of a graph to the minimum and maximum numerical values of a set of data.

Bad Block
A damaged block existing on your volume that the system cannot access. Blocks become damaged from wear or abuse.

BASIC
Beginner's All-purpose Symbolic Instruction Code; BASIC is an interactive, "algebraic" type of computer language developed at Dartmouth College, Hanover, N.H., by Kemeny and Kurz that combines English words and decimal numbers. It is a widely available, standardized, simple beginner's language capable of handling industrial, laboratory, and business applications. MINC BASIC is a special version of this standard language.

Binary
The number system with a base of two, used by the internal logic of most digital computers.

Block
A unit of space, or capacity, on a storage volume. A block can hold 512 characters.

Branching
A programming technique that transfers control to a program statement independent of its sequence in the program. Branching allows you to construct programs containing specialized parts that execute only if a particular condition exists.

Brand
A small vertical line on a graph used to mark the exact horizontal location of data.

Bug
Any error existing in a program.

Chain
A connected series of separate programs or program parts. Chaining is the simplest way in MiniMINC to prepare separate programs for parts of a complex task and then run them together.

Channel
A logical pathway used in MINC BASIC for the sending and receiving of data. There are 12 MiniMINC file channels available through the OPEN statement that allow programs to access files for data.

Character mode
The stylistic features shared by characters appearing on a terminal screen. MiniMINC character modes consist of boldface, underline, reverse video, flashing, or normal elements. You can specify character modes alone or in combination.

Column
The vertical position on the MiniMINC terminal screen in which a character is displayed. Compare with Row.

Command
A word, mnemonic, or character which, by virtue of its syntax and its position in an input line, causes a computer to perform a predefined operation. In MINC BASIC, commands can be used only in immediate mode.

Compile
To convert an ASCII file into its smaller binary form. A compiled MINC BASIC program requires less storage space and less time for handling by MiniMINC.

Computer
An electronic device that stores, retrieves, and processes data through the execution of programmed instructions.

Computer system
A combination of devices, programs, and documentation designed for automatic problem solving following or concurrent with human intervention.

Concatenation
The joining of two strings of characters to produce a longer string.

Conditional branch
Branching within a program that involves testing for a condition before executing some statement or series of statements.

Conditional transfer
The action resulting from a conditional branch.

Constant
An invariable and unchanging value. Compare with variable.

Control character
The product of pressing the CTRL key and a letter key on the MiniMINC keyboard simultaneously. A control character sometimes appears on the terminal screen as a caret (^) next to the appropriate letter.

Control command
A command that alters a current user or computer activity. Unlike other commands that work only when READY is displayed on the screen, you can give a control command at almost any time to establish your "control" over the computer.

Cursor
A unique symbol that marks your current position on a video terminal screen. Whatever character you type will appear at the cursor. With the MiniMINC terminal, you can use either a flashing

box or a flashing underscore as the cursor symbol.

Data

A term used to denote any or all facts, numbers, letters, and symbols forming basic elements of information that can be processed by a computer.

Debug

To detect, locate, and correct errors in a program.

Default

An option or argument value assumed by a program if a specific value is not supplied by the user.

Define

To assign a value to a variable or to assign an expression to a user function.

Device

A piece of equipment such as a line printer or diskette drive unit.

Digital

A term referring to numbers represented as discrete units. A digital computer interprets electronic signals as discrete digits (0 or 1), while an analog computer interprets them as analogs of a wider range of numbers.

Directory

A table that contains the name, size, and location of every file on a mass-storage volume.

Diskette

A thin, flexible, magnetic, oxide-coated disk resembling a 45 rpm record and permanently enclosed in an 8-inch square envelope. Diskettes are random-access, mass-storage volumes.

Display mode

An attribute of the entire terminal screen. MiniMINC display modes determine the background color of the screen, the width of character lines, and the scrolling method.

Distribution medium

Any volume or material used in distributing copies of system programs. Distribution media are generally diskettes or magnetic tape.

Drive

A unit that holds and manipulates a mass-storage medium such as a diskette or magnetic tape.

Dummy argument

An argument inside the definition of a function that holds the place for the formal argument.

E notation

Method of representing numbers in MiniMINC greater than 999999 or less than .01. The letter E appears between the mathematical expression and the exponent in lieu of the "x10" used in scientific notation. $1.0\text{x}10^{-4}$ becomes 1.0E-4 in E notation.

Edit
To arrange and/or modify the format of data and programs, including insertion and deletion of text.

Editor
The program you use to create and edit files within a computer. Editors manipulate characters without regard for their significance.

Element
The smallest unit of storage within an array. For example, N(12) represents an element in the array N.

Error message
A brief message MiniMINC displays on your terminal when an error occurs. Error messages identify the nature of user and computer errors and sometimes suggest a corrective action.

Execute
To carry out an instruction or run a program on the computer.

Expression
A combination of variables, constants, and operators (as in a mathematical expression).

Field
A specified area of a mass-storage volume, statement, command, or terminal screen used for a particular category of data.

File
A logical collection of data treated as a unit, which occupies one or more blocks on a mass-storage volume, and has an associated file name and type.

File allocation
The MiniMINC practice that determines the placement of a file on a diskette. The MiniMINC system stores a file in the first available area that is large enough to hold the file. In some cases, MINC BASIC requires that the available space be twice as large as the file before allocating space.

File Number
A channel number associated with a file that you use when referring to the file within a program.

File spec
See File specification.

File specification
The part of a command or statement that identifies a file's name and type and the device where it can be found.

File Type
The one-to-three character string assigned to a file either by the user or the computer. The file type follows the file name and is preceded by a period. The file type serves to further identify the kind of file as in PRGRAM.BAS where PRGRAM is a BASIC program.

Flow (of control)
The order in which a program's statements are executed.

Form Feed
An ASCII character that delimits pages within a file. On some printing equipment, a Form Feed causes a skip to the next page.

Formatting
An ungrammatical term that has come to mean the recording of timing information on magnetic, mass-storage volumes. The timing information is necessary for drive units to be able to find information stored on the volumes.

Function
A subprogram that performs commonly used operations (for example, the square root calculation function). The name of the function takes on the value of the calculation and can be used as a program variable. Some functions exist as part of MiniMINC and others can be user-defined.

Graph number
A label (0,1, or 2) used in graphic routines to specify each graph individually or both in a two-graph display.

Graph region
The portion of a terminal screen reserved for graphic display. MiniMINC has three graph regions: upper, lower, and full screen.

Graphic memory
The memory area of a MiniMINC terminal that preserves graphic data. The graphic memory allows you to display the same data in more than one graphic format.

Graphic program
Programs that compute data and construct diagrams for the display of that data on a graphic display terminal.

Graphic routine
A statement that performs a routine task associated with graphic display.

High-level language
A programming language (for example, BASIC, FORTRAN, and COBOL) whose statements are problem-oriented and typically translated into more than one machine language instruction.

Immediate mode
The mode in which a computer executes statements as soon as you type them, without requiring a program.

Infinite loop
A program segment that, when executed, repeats endlessly because it lacks any provision for termination.

Initialize
To prepare a new mass-storage volume for use with a computer or to erase a used volume in preparation for reuse. Initialization sets up an empty file directory on a volume.

Input
The act of placing information inside a computer, or the information itself.

Integer variable
A variable representing only whole numbers within some range (for example, -32768 to +32767).

Intelligence
The capacity of a machine for mimicking activity, such as reasoning or learning, normally attributed to human intelligence.

Interlacing
A feature of the MiniMINC terminal. Interlacing is the placing of additional scan lines between the normal ones in the terminal's screen. This increases character density by reducing the amount of undefined space within each character. Interlacing is useful whenever you photograph the screen.

Left-justified
An arrangement of text where every line begins at the same distance from the lefthand margin.

Line
A string of characters terminated with a vertical tab, form feed, or line feed/carriage return combination.

Line printer
A printing device that composes a line of characters before printing it. The term is sometimes erroneously used to mean a character printer.

Literal
A language element that permits the explicit representation of values in expressions. In most languages, a character string literal is enclosed in single or double quotes, while a numeric is not. In either case, the value is what it appears to be, not the name of a value.

Load
1. To insert a program or data into memory.
2. To place a removable disk in a disk drive and start the drive.

Logical expression
An expression having one of two possible values—true or false.

Loop
A sequence of programming statements that, when executed, repeat continuously until an end condition is met.

Machine language
The binary language used by a computer when performing operations.

Mass storage
Pertaining to a volume that can store large amounts of data readily accessible to the computer.

Master volume
A single-purpose, mass-storage volume dedicated to the creation of system, demonstration, and user volumes.

Memory
The part of a computer that temporarily stores programs and data

intended for immediate use.

Mixed mode
Arithmetic operations involving a combination of integer and real operands.

Modem
A modulator/demodulator apparatus that permits long distance transmission of digital data.

Multiple branch
A program statement that transfers program control to one of several possible locations.

Multiway branch
Same as a multiple branch.

Nesting
Including an executable operation within a larger one of the same or similar type. BASIC allows you to override operator priorities (^ , * , / , + , - descending order from left to right) by nesting expressions with parentheses. BASIC also allows nesting of programming loops, where the internal loop must be totally contained within the external loop.

Nonprogram file
A sequential or virtual array file containing anything but a program.

Nonsystem volume
A volume capable of storing programs and data, but lacking the structure and programs necessary for starting and maintaining the system.

Null
An ASCII character whose numeric code is 0.

Numeric variable
A computer storage location reserved for a numeric value.

Octal
Pertaining to the number system with a base of 8.

One-dimensional array
An array with one subscript.

Operation
The act specified by a single computer instruction. A program step undertaken or executed by a computer (addition, comparison, and multiplication, for example).

Option
An element of a command, command string, statement, or routine that enables you to select from among several associated alternatives.

Output
1. The result of a process.
2. The transfer of data from internal to external storage or to the terminal screen.

Overlay
The practice of merging a program segment with statements existing apart from that segment.

Owner name
Name indicating ownership of a volume. The user assigns this name to a volume during its initialization or its adaptation by a master volume. The owner name appears whenever the user requests a volume's directory.

Page
That portion of a text file delimited by form feed characters and generally 50-60 lines long.

Plotter
A device using an automatic pen or pencil to construct visual representations of data. Plotters sometimes receive plotting coordinates from digital computers.

Point
A single data value.

Precision
The resolution of a number's representation by a computer. MiniMINC's precision extends to six significant digits for real numbers and four (plus) digits for integers.

Print zone
An assumed horizontal subdivision of the terminal screen. MiniMINC acts as though the terminal's screen is divided into five print zones of 14 columns each, skipping to the next zone whenever it encounters a comma in a PRINT statement.

Program
A set of computer instructions or symbolic statements combined to perform some task.

Program file
A program stored on a volume. A program file has a file name and type.

Program flow
The order in which a program's statements are executed. Also known as flow of control.

Programming language
A computer-oriented language used in writing programs.

Prompt
A word or message printed by the system that requests or suggests some action on your part.

Random access
Access to data in which the next location from which data is to be obtained does not depend on the location of the previously obtained data. Diskettes are examples of random access storage media. Contrast with Sequential access.

Raster unit
A fixed unit of linear measure used in determining bar width for

MiniMINC bargraphs. The MiniMINC terminal screen measures 240x512 raster units.

Read

To copy data from one form of storage to another, usually from an external device to internal storage such as memory.

Real number

Commonly understood to be any positive or negative number, excluding imaginary or complex numbers. However, computers impose limits on the range (approximately $.29\text{x}10^{-38}$ through $1.7\text{x}10^{+38}$) and precision of real numbers. Consequently, all irrational and many rational numbers are excluded from the working definition of a real number.

Real variable

A variable for real numbers.

Relational operator

A symbol representing the relationship of one value to another. The relational operators in MINC BASIC are = (equal to), < (less than), < = (less than or equal to), > (greater than), >= (greater than or equal to), and <> (not equal to).

Right-justified

An arrangement of text where every line ends at the same distance from the right-hand margin.

Routine

A statement that performs some routine task. Routines return their results through arguments, unlike functions that return values within their names.

Row

The horizontal strip on the terminal screen in which a line of text can be displayed. The MiniMINC terminal has 24 rows. Text coordinates are given in rows and columns, whereas graph coordinates use X and Y.

Run

To execute a program.

Save

To store a program or data as a file on a volume. Information not saved is erased when you shut off the computer or use MiniMINC's workspace for another operation.

Scientific notation

Method of representing real numbers as a number from 1-10 multiplied by an appropriate power of 10. For example, 56930 becomes $5.693\text{x}10^4$ in scientific notation. See E notation.

Scrolling

The upward movement of data on a terminal screen to accommodate new data. The oldest line of data disappears from view at the top of the screen when the new line appears at the bottom.

Scrolling area

The region on a terminal screen where scrolling takes place. The scrolling area normally occupies 24 rows.

Search
In MiniMINC, the keypad editor's examination of a file or part of a file for a designated string.

Search model
The string you type as the object of a search.

Sequential access
Access to data in which the next location from which data is to be obtained sequentially follows the location of the previously obtained data. Contrast with Random access.

Sequential file
A file that must be accessed sequentially. MiniMINC looks for information in a sequential file by checking one position after another in a linear direction.

Serial
A mode of data transmission in which the components of each data word are transmitted according to a prescribed protocol, one after another (serially) along a single pair of lines from a sending to a receiving device.

Serial line unit
One of six ports on the back of the chassis dedicated to hookups with serial line devices. Such devices include your terminal and printers. Often referred to as an SLU.

SLU
Same as a Serial line unit.

Statement
An element of a program. Programs are sequential arrangements of statements.

Storage medium
Any type of sequential-access or random-access volume used for storing files.

String
A connected sequence of characters.

String variable
A variable used for any ordered collection of numeric, alphabetic, and special characters up to 255 in length.

Strip chart mode
The dynamic display of data in graphic form. This MiniMINC feature lets you see a large number of points plotted in a continuous flow across the screen, resembling the action of a strip chart. Also called move mode.

Subroutine
A group of statements arranged within a program so that program control can pass to the subroutine and back to the program again. Subroutines usually perform tasks required more than once by the program.

Subscript
A number appended to a variable name to uniquely identify specific elements of an array. Subscripts are enclosed in parentheses.

Substring
Any contiguous part of a larger string.

System file
Any one of a group of files containing the programs and data that, together with the equipment and documentation, comprise a computer system.

System volume
The storage medium that contains the system files.

Syntax
The structure of expressions in a language and the rules governing that structure.

Terminal
The primary communication device between you and the computer system. A terminal has a keyboard and a display mechanism.

Timesharing
A method of allocating resources to multiple users so that the computer, in effect, processes a number of programs concurrently.

Tone
The sound signal emitted by the MiniMINC terminal whenever an editor operation fails or you type to the end of a line.

Two-dimensional array
An array requiring two subscripts (commonly portrayed as a matrix).

Unconditional branch
A programming technique that transfers program control to the statement you specify without regard for sequence.

Unconditional transfer
Same as Unconditional branch.

User volume
The storage medium reserved for user files and the HELP text file.

Variable
1. A computer storage location that can contain a value that changes during computation.
2. The symbolic representation of a variable (for example, F, F%, and F$).

Virtual array file
A file containing information directly accessible by MiniMINC. The system can retrieve any piece of data without first examining the information preceding it. The file is virtual because it appears in the program as an array even though it is a file stored on a volume.

Vol id
See Volume identifier.

Volume
A logical storage medium — but not any particular one. A generic term used for general reference.

Volume identifier (Vol id)
Identifying name assigned to a volume by a user during the volume's initialization or its adaptation by a master volume. The volume identifier appears whenever the user requests a volume's directory.

Window
The relationship for a single graph that defines placement of coordinates at physical screen positions, so called because the graph is like a window through which you view a portion of the X-Y coordinate space.

Word
The number of characters treated as a unit in the workspace. MiniMINC words can contain 2 characters, 1 integer, or one half of a real number.

Workspace
MiniMINC's storage area for temporarily holding a program you type and its associated values.

Wrap symbol
The two-character symbol displayed by the keypad editor whenever a file line is longer than a screen row. Indicates that the preceding line does not end with a line terminator.

Wrapped line
Any line that is continued onto the next row because it is too long to fit in a single row.

Write
To copy data from internal storage to an external device or to insert data into a storage location.

INDEX

Introduction to MiniMINC
AA-H400A-TC

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require an immediate answer and you are under warranty, call the appropriate MINC Customer Support Center. (The MINC Customer Support Centers are listed in the MINC Newsletter.)

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

Did you find errors in this manual? If so, specify the error and the page number.

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify)

Name Date

Organization

Street Telephone

City State Zip Code or Country

--------- Do Not Tear – Fold Here and Tape ---------------------------------

No Postage
Necessary
if Mailed in the
United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO. 33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SOFTWARE PUBLICATIONS
200 FOREST STREET MR1-2/E37
MARLBOROUGH, MASSACHUSETTS 01752

-------- Do Not Tear – Fold Here and Tape ---------------------------------