# MiniMINC

## MiniMINC Supplement

**August 1979**

This document contains supplementary information on MiniMINC routines and devices. The information in the *MiniMINC Supplement* is not documented in the other MiniMINC manuals.

Order Number AA-H789A-TC

MiniMINC

VERSION 1.1

Software and manuals should be ordered by title and order number. In the United States, send orders to the nearest distribution center. Outside the United States, orders should be directed to the nearest DIGITAL Field Sales Office or representative.

NORTHEAST/MID-ATLANTIC REGION

Technical Documentation Center
Cotton Road
Nashua, NH 03060
Telephone: (800) 258-1710
New Hampshire residents: (603) 884-6660

CENTRAL REGION

Technical Documentation Center
1050 East Remington Road
Schaumburg, Illinois 60195
Telephone: (312) 640-5612

WESTERN REGION

Technical Documentation Center
2525 Augustine Drive
Santa Clara, California 95051
Telephone: (408) 984-0200

**digital equipment corporation • marlboro, massachusetts**

First Printing, August 1979



The postage-prepaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist us in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

| | | |
|---|---|---|
| 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 |

9/79-14

# CONTENTS

## FIGURES

## TABLES

# PREFACE

This supplement describes certain features of the MiniMINC system that are not documented in other MiniMINC manuals. It contains material on new graphic routines, serial ASCII programming, and on the connection of serial ASCII devices to a MiniMINC system.

The supplement also supplies troubleshooting information for MiniMINC devices.

Most of the features described in this supplement apply to graphic programming and to the use of the serial ASCII routines. Before reading this supplement, you should understand general properties of the MiniMINC system, including the use of MINC routines in programs.

Therefore, you should read this supplement after you have read Books 1-4 of your MiniMINC documentation set.

# CHAPTER 1
# SERIAL ASCII PROGRAMMING

## GENERAL DISCUSSION

Serial ASCII programs allow a MiniMINC system to communicate with serial ASCII instruments. These instruments respond to ASCII characters received on a communication line. Many such instruments can also return information to the MiniMINC system, and this information is also transmitted as a series of ASCII characters.

Serial ASCII instruments are used in a very wide range of applications and include such devices as interactive plotters and analytical balances. Not all instruments interpret particular ASCII characters the same way. The character LF (ASCII 15) is commonly used by line printers and terminals to control the movement of paper; on other, unrelated devices, this character might be used to activate a different function or it might be transmitted from the device to MiniMINC to indicate a condition peculiar to that device. A particular device's interpretation of characters forms the "protocol" for that device.

Because individual serial ASCII instruments have device-specific protocols, this manual cannot generalize about device protocols. Each type of instrument comes with its own user's manual, which defines the instrument's protocol.

The section "Working with Serial ASCII Apparatus" describes the physical connection of serial ASCII devices to a MiniMINC system. The only general requirement for connecting such a device to MiniMINC is that the device conform to the RS-232C standard for serial interfaces. The specifications and user documentation for serial ASCII devices will state explicitly whether

the device conforms to RS-232C. If a device conforms to this standard, "Working with Serial ASCII Apparatus" gives you specific procedures for connecting the device to MiniMINC.

As many as three serial ASCII devices can be connected simultaneously to a MiniMINC.

After you have connected a device to your MiniMINC system and have learned the device protocol, you can write serial ASCII programs with a set of special MiniMINC routines:

1. SET_SERIAL specifies the speed at which characters are transmitted to or from a particular device, the number of bits per character, and the parity scheme (ODD, EVEN, or NONE) used by the device to verify received characters. SET_SERIAL also defines the speed at which characters can be printed by a line printer device (LP:) connected to the MiniMINC.

2. CIN receives characters from a connected serial device.

3. COUT sends characters to a connected serial device.

**NOTE**

In setting transmission speeds (baud rates), the SET_SERIAL routine accepts baud rate parameters as high as 19200. However, as discussed in the section "Working with Serial ASCII Apparatus," the maximum speed at which characters move through the three serial ports is 240 characters per second.

A high baud rate, such as 9600, may be required to make an electronically valid connection to certain devices. In such a case, the SET_SERIAL argument must be 9600, and the signals that represent an individual character will be sent at that rate. However, no more than 240 of such characters can be sent in a second.

Therefore, baud rates do not translate consistently into a number of characters transmitted per second. A baud rate of 300 means that 30 characters are transmitted per second *while transmission is in progress*. If characters are transmitted in bursts, and not continuously, the net transfer rate (or "throughput") will be lower.

## SET_SERIAL: SET ATTRIBUTES OF SERIAL LINE

### Operation

Some serial ASCII instruments require a specifc set of attributes for their physical communication line to a computer. A familiar example of of such an attribute is the baud rate.

SET_SERIAL is used to describe to the MiniMINC system the attributes of a particular serial line. The serial line ports are labeled TERMINAL #1, TERMINAL #2, and TERMINAL #3 on the back of your MiniMINC chassis.

You can also use SET_SERIAL to set the baud rate and parity of the serial port labeled PRINTER. Depending on the type of output device you connect to the PRINTER port, you may need to change the port's baud rate from its default speed of 300 baud.

A single SET_SERIAL statement sets the attributes of a single serial port. To set attributes for several ports, use an equal number of SET_SERIAL statements.

### Statement Form

SET_SERIAL

### Argument Descriptions

**option** This argument is a string expression consisting of a single option word enclosed in quotation marks, or consisting of several option words separated by commas, with the whole series enclosed in quotes. The valid option words are as follows.

1. If you omit the option string, NONE and LP: are selected by default.

   If you include a port-designator and omit the option string, the default becomes simply NONE.

   LP: directs SET_SERIAL to the PRINTER port. If you specify LP:, you cannot specify a port-designator argument, because there is only one PRINTER port.

2. ODD sets the designated port to odd parity. Some instruments require characters to be transmitted and received with odd parity.

3. EVEN sets the designated port to even parity.

4. NONE sets the designated port to no parity.

**port-designator** This is an integer from 1 to 3, specifying a single port from the group labeled TERMINAL #1, TERMINAL #2, and TERMINAL #3. Recall that you cannot specify a port-

designator and the LP: option in the same SET_SERIAL statement.

If you omit the port-designator argument, a value of 1 is used by default.

**baud-rate** This is the baud rate at which the designated serial port will operate. The only valid values are 50, 75, 110, 134.5, 150, 300, 600, 1200, 1800, 2000, 2400, 3600, 4800, 7200, 9600, and 19200.

If you omit the baud-rate argument, a rate of 300 baud is set by default.

**bits** This gives the number of bits per character. The only valid values are 5, 6, 7, and 8. Note that this argument specifies the number of bits actually used to represent the character code.

If you omit the bits argument, the value 8 is used by default.

**NOTE**

At 110 baud, an additional two bits are appended to the character code as stop bits. At all other baud rates, one bit is used as a stop bit. Character codes are also preceded by a single start bit. Consequently, characters are ordinarily represented by 10 bits in a digital stream (1 start bit + 8-bit character code + 1 stop bit). This information may be useful in calculating the character throughput for a particular application. See the discussion of the CIN routine for further information.

**Related Routines**

1. CIN and COUT transmit characters through the TERMINAL ports. Use SET_SERIAL to designate the port's attributes before using CIN or COUT.

2. Because SET_SERIAL can change the baud rate of the PRINTER port, it can affect the speed with which information is printed. Commands that send information to the printer include SAVE, COPY, and DIR. The OPEN statement can also open a file called "LP:". Information subsequently output to the file "LP:" is directed to the PRINTER port and will be printed on a line printer if one is connected. For more information on SAVE, COPY, DIR, and OPEN, see *Book 3: MINC Programming Reference*. Further in-

formation on MiniMINC line printers is provided in this manual.

**Restrictions**

1. All of MiniMINC's serial ports have physical limitations on transmission speed. For instance, the three TERMINAL ports have a maximum character throughput of 240 characters per second. However, to form an electronically valid connection between MiniMINC and a particular instrument, you may need to specify, with SET_SERIAL, a high baud rate such as 9600.

   A table of the maximum character rates is given in the section "Working with Serial ASCII Apparatus."

**Errors**

?MINC-F-Invalid or conflicting options requested
You included some option word other than LP:, EVEN, ODD, or NONE. Reenter the SET_SERIAL statement with valid options.

?MINC-F-Serial line number must be in range 1-3
You entered a port-designator that was neither 1, 2, nor 3. Reenter the SET_SERIAL statement with a valid port-designator.

?MINC-F-Illegal baud rate specified
You entered a baud-rate that was not a valid value. Reenter the statement with a valid baud rate.

?MINC-F-Illegal number of bits specified
You entered a bits argument that was neither 5, 6, 7, nor 8. Reenter the statement with a valid value.

?MINC-F-Specify either LP: or serial line number, but not both
You specified the LP: option and a port-designator in the same statement. Reenter the statement. If you want to set attributes for a printer, omit the port-designator. If you want to set attributes for TERMINAL #1, TERMINAL #2, or TERMINAL #3, omit the LP: option; you may also need to specify the ODD or EVEN option in this case.

**Examples**

SET_SERIAL("LP:",,1200)
sets PRINTER port to operate with a 1200-baud printer (no parity).

SET_SERIAL("LP:")
resets PRINTER port to 300 baud.

SET_SERIAL
same as SET_SERIAL("LP:")

last CIN statement, CIN retrieves 128 characters and then halts with a fatal error.

If you omit the option word, the first character assigned to the string is the first character that arrives after the current CIN statement begins executing. Characters arriving prior to the current CIN statement are lost.

In RETRIEVE mode, CIN collects as many characters as specified by the string-length argument. For example, if the string length is 10, then CIN collects 10 characters starting with the first character after the most recent CIN statement. If more than 10 characters have already arrived, CIN still collects only 10 characters; the surplus characters can be retrieved by the next CIN statement. If fewer than 10 characters have arrived, CIN collects all the characters and waits for the remaining characters necessary to make a string of 10.

When you specify a variable-length string (string-length=0) in RETRIEVE mode, CIN continues collecting characters until one of the following conditions occurs:

- A carriage-return character (ASCII 13) is received.
- The timeout interval expires.
- The MINC BASIC string limit of 255 characters is reached.
- The limit of 128 previous characters is reached (error condition occurring when more than 128 characters have arrived since the last "non-RETRIEVE" CIN statement).

**string-name** The name of the string variable to which collected characters are assigned. This argument must be either a string variable, such as S$, or an element of a string array, such as S$(1).

You cannot collect an array of strings with a single CIN statement.

You cannot omit the string-name argument from a CIN statement.

**string-length** The length of the character string to be collected. Valid values are 0, and the range of integers from 1 to

255. The use of RETRIEVE mode imposes certain restrictions on the length of a received string. See "Restrictions" for this routine.

A value of 0 specifies a variable string length. For variable strings, CIN collects characters until the timeout interval is reached, or the maximum of 255 characters is reached, or the carriage return character is received. The carriage return character itself does not become part of the collected string, nor is it included in the character count. Note that if your instrument terminates its messages with a carriage return and a line feed, your program will always see the line feed as the beginning of the next message.

Values in the range 1-255 specify fixed-length strings. When collecting characters for a fixed-length string, CIN recognizes no particular character as a terminator. Therefore, such characters as the carriage return can become part of the collected string and therefore are included in the character count.

The timeout interval is especially useful in collecting fixed-length strings. When collecting characters for a fixed-length string, CIN continues executing until the number of requested characters has arrived and been collected. Therefore, the timeout interval assures that CIN finishes executing in case, for some reason, the routine never receives a sufficient number of characters.

**port** The serial port carrying input characters. Valid values are 1, 2, and 3, designating the ports labeled TERMINAL #1, TERMINAL #2, and TERMINAL #3, respectively.

Before using CIN, be sure to use the SET_SERIAL routine to select the correct attributes for the designated port.

If you omit the port argument, a value of 1 (designating TERMINAL #1) is used by default.

**timeout-interval** The time interval after which CIN considers the current string to be complete. Valid values are 0 and the range of decimal numbers greater than or equal to 0.1.

A value of 0 specifies no timeout.

A value greater than or equal to .1 specifies the timeout interval in seconds. The interval .1 seconds is the smallest interval

guaranteed to be precise. The resolution of the timeout interval is one tick of the system clock.

If you omit the timeout-interval argument, a value of 0 (no timeout) is used by default.

**Related Routines**

1. COUT sends character strings to an instrument connected to a serial port. COUT can send strings of fixed or variable length.

2. SET_SERIAL defines the attributes of a particular serial port. These attributes are the baud rate, the number of bits per character, and the type of parity (ODD, EVEN, or NONE). These attributes must be set to match the specifications of a particular instrument.

   In a program that transfers serial ASCII characters, the attributes of the serial port must be set *before* a CIN statement attempts to receive characters from the port. In such a program, be sure that the SET_SERIAL statement is executed before any CIN or COUT statement.

   SET_SERIAL has default values for all its arguments. However, it is good practice to include the required values explicitly in the SET_SERIAL statement. If you include the values explicitly, the resulting program will be easier for you and other users to interpret and modify.

   Include a SET_SERIAL statement in any program that uses CIN and/or COUT. Do not assume that the attributes of a port are "already" set to the correct values, even if you have set the attributes in a previous program.

3. The SAVE, OPEN, and COPY statements allow you to list programs or print data on a line printer. In these statements, the line printer is identified by the logical name LP:. To work with these statements and with the logical name LP:, the line printer must be connected to the PRINTER port. CIN and COUT cannot send characters to the PRINTER port.

   SAVE, OPEN, and COPY are related to CIN and COUT in that they, too, allow the serial transfer of ASCII characters to a connected device. However, they cover the special case of line printers. CIN and COUT should be used instead for other types of serial transfer.

**Restrictions**

1. A CIN statement with the RETRIEVE option can retrieve up to 128 characters received since the completion of the most recent CIN statement. Therefore, you can use CIN to collect an essentially continuous stream of character input because you can process strings as they arrive without losing intervening characters.

   The 128-character maximum also imposes a practical limit on the sustainable rate at which characters can be input. At 9600 baud, 128 characters can arrive in approximately 135 milliseconds. At 1200 baud, 128 characters can arrive in approximately 1 second.

   To establish the actual character throughput for a given application, divide the baud rate by the number of bits per character plus stop bits (usually 10)—this gives the number of characters that the TERMINAL port can handle per second ("hardware" throughput). Figure the time taken by your program to process 128 characters and divide the time by 128—this gives the number of characters that the program can handle per second (program throughput). The character throughput is the hardware throughput, program throughput, or 240, whichever is lowest.

2. CIN does not echo received characters. That is, CIN does not automatically transmit a received character back to the sender for display or verification.

3. If you direct a CIN statement to a TERMINAL port to which no device is connected, the system may appear to be malfunctioning or inactive. Type two CTRL/C commands to exit from the CIN statement.

**Errors**

?MINC-F-Channel or unit # not in system for the routine
?MINC-F-Data lost—transfer rate too high
You specified an invalid channel number (the port argument) in the statement.

More than 128 characters have arrived since the last CIN statement finished executing (RETRIEVE mode).

The characters are arriving too quickly for CIN to receive them.

Characters are arriving at the wrong baud rate. (Be sure that you have set the baud rate correctly with a SET_SERIAL statement.)

?MINC-F-Invalid or conflicting options requested
You entered an invalid option word in the CIN statement. If you include an option word, RETRIEVE is the only valid option.

?MINC-F-No workspace available for the string specified
The string you specified in the string-name argument requires more memory than is available in the workspace. Reduce the size of the program if possible, or use the EXTRA_SPACE command.

?MINC-F-Use array element instead of array for argument #
You specified an array name for the string-name argument. The string-length argument can accept input for a single string only; therefore, the string-length argument must be either the name of a single string or the name of an *element* of a string array, such as C$(1).

**Examples**

None.

## COUT: SEND CHARACTER STRING OUTPUT

COUT sends characters in serial ASCII format via a serial transfer channel.

**Operation**

MiniMINC systems have five serial ASCII channels. The ports to the channels are labeled CONSOLE, PRINTER, TERMINAL #1, TERMINAL #2, and TERMINAL #3 on the back of the MiniMINC chassis. The chassis also has a port called MODEM that is not used by MiniMINC. The MODEM port is used by optional communication software that is not discussed in the MiniMINC manuals.

The CONSOLE and PRINTER ports are reserved for use by the MINC VT105 terminal and by line printers, respectively. COUT sends characters to instruments via the TERMINAL ports.

See section "Working with Serial ASCII Apparatus" for instructions on how to connect serial ASCII devices to the TERMINAL ports.

**Statement Form**

COUT(option,string-name,string-length,port)

**Argument Descriptions**

**option** The character string selecting an optional mode for COUT.

If you include the option word WAIT, COUT pauses and waits for the transfer to finish before allowing the next program statement to execute.

If you do not include an option word, COUT starts the transfer and passes control to the next statement when at most 32 characters remain to be sent. In this mode, the transfer will finish while the program continues.

**string-name** The variable name for the data sent. COUT requires that the string-name argument be a string expression. You cannot send a complete string array with a single COUT statement.

You cannot omit the string-name argument from a COUT statement.

**string-length** The number of characters to send. Valid values are 0 and integers in the range 1-255.

A value of 0 sends as many characters as are contained in the string, up to the string maximum of 255 characters. COUT appends a carriage return to the transmitted string when the string-length is 0.

Values in the range 1-255 send a fixed number of characters from the string; no carriage return is appended. The transmission begins with the first character in the string. If the string does not contain enough characters, COUT sends only as many as are in the string.

If you omit the string-length argument, a value of 0 (variable-length string) is used by default.

**port** The port to the serial channel that carries output characters. Valid values are 1, 2, and 3, designating the ports labeled TERMINAL #1, TERMINAL #2, and TERMINAL #3, respectively.

Be sure that the port you designate has been assigned the correct attributes by a SET_SERIAL statement.

If you omit the port argument, TERMINAL 1 is designated by default.

**Related Routines**

1. CIN collects character strings from an instrument connected to a serial channel. CIN can receive strings of fixed or variable length.

2. SET_SERIAL defines the attributes for a serial port that is then designated in a COUT statement. When you use COUT

in a program, include the appropriate SET_SERIAL statement, and be sure that SET_SERIAL is executed before COUT. Do not assume that the designated port is "already" set to the required attributes, even if you have done so in a previous program.

3. The SAVE, OPEN, and COPY statements allow you to list programs or print data on a line printer. In these statements, the line printer is identified by the logical name LP:. To work with these statements and with the logical name LP:, the line printer must be connected to the PRINTER port. CIN and COUT cannot send characters to the PRINTER port.

   SAVE, OPEN, and COPY are related to CIN and COUT in that they also allow the serial transfer of ASCII characters to a connected device. However, they cover the special case of line printers. CIN and COUT should be used instead for other types of serial transfer.

**Restrictions**

1. When you omit the option word from a COUT statement, COUT can finish executing when up to 32 characters remain to be transmitted. A second COUT statement, again without the option word, can start executing before the first transfer is complete. However, the second statement cannot finish executing until the total number of characters remaining from both statements is less than or equal to 32.

**Errors**

?MINC-F-Channel or unit # not in system for the routine
You specified an invalid channel number (the port argument) in the routine statement. The port argument must have the value 1, 2, or 3.

?MINC-F-Invalid or conflicting options requested
You entered an invalid option word in the COUT statement. If you include the option word, WAIT is the only valid choice.

?MINC-F-Use array element instead of array for argument #
The string-name must be a string expression.

?MINC-F-Serial output channel # is not ready.
You specified a channel (port) that was not ready, such as a port to which no device is connected. This message may also occur if a connected device is not asserting the required DTR (Data Terminal Ready) signal.

**Examples**

None.

## WORKING WITH SERIAL ASCII APPARATUS

All communication between the MiniMINC system and external apparatus occurs through the serial input/output ports on the rear of the MiniMINC chassis (see Figure 1). These ports provide EIA standard signals and are not suitable for communicating with ASR 33/35 teletypewriters and other equipment requiring 20 milliampere current loop interfaces.

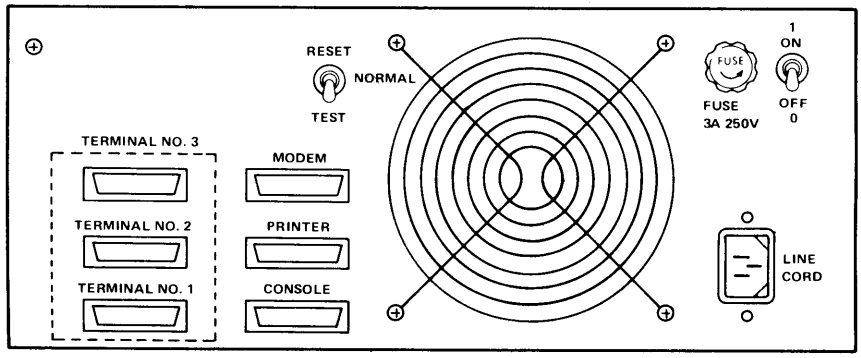



Figure 1. MiniMinc Serial Input/Output Ports

One of these ports, labeled MODEM, provides all the control lines necessary to transmit and receive data via modulator/demodulator apparatus (modems) of the kind provided by common carriers and others for long distance transmission of digital data.

**NOTE**

The MODEM port is not used by the MiniMINC system. It cannot be addressed by the routines SET_SERIAL, CIN, and COUT. The MODEM port is used by optional communication software that is not discussed in the MiniMINC manuals.

A second port, CONSOLE, is dedicated to the MiniMINC terminal.

The remaining four ports (PRINTER, TERMINAL #1, TERMINAL #2, and TERMINAL #3) are functionally identical except for their maximum operating speeds and are capable of communicating with most terminals, printers, and other apparatus designed to operate with serial ASCII code. The speed of all input/output ports (except MODEM and CONSOLE) can be set to any permitted rate with the SET_SERIAL routine.

The serial input/output ports are capable of communicating in either full- or half-duplex mode at the rates specified in Table 1. Note, however, that each rate assumes no competition from other devices simulataneously communicating with the system. The actual maximum rates for all devices on a system will depend on how many other devices are competing for attention, how fast the other devices are operating, what mode they are operating in (half or full duplex), whether they operate at a continuous pace or in bursts, and so on.

Table 1. Maximum Character Rates for MiniMINC Serial I/O Ports

| *I/O Port* | *Maximum Rate (char/sec)* |
|---|---|
| Console | 960 |
| Printer | 960 |
| Terminal 1 | 240 |
| Terminal 2 | 240 |
| Terminal 3 | 240 |

When printers and terminals manufactured by Digital Equipment Corporation are involved, no special cables are required. If the cable provided with the unit mates with the desired input/output port, it should make the necessary connections without further user manipulation. When apparatus not manufactured by Digital is to be used, it may be necessary to make up custom cables. Figure 2 and Table 2 provide the required information.

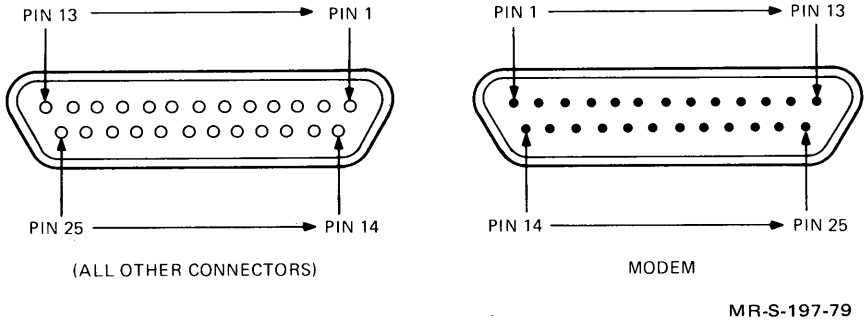



Figure 2. Serial Input/Output Connector Pin Identification

Table 2. MiniMINC Serial Input/Output Port Connector Signals

| *Connector* | *Signal Designation* | *Device Group* |
|---|---|---|
| Console (female) | | |
| | | LA38 DECwriter IV, |
| pin 2 | Term Xmit Data | LA34 DECwriter IV, |
| pin 3 | Term RCV Data | LA36 DECwriter II, |
| pin 20 | Term RDY | VT50 Decscope, |
| pin 1 | Chassis GND | VT100 video terminal, |
| pin 7 | Signal GND | VT105 graphic terminal, |
| | | VT52 DECscope |
| Printer (female) | | |
| pin 2 | LP Xmit Data | LA35 DECwriter II, |
| pin 3 | LP RCV Data | LA36 DECwriter II, |
| pin 20 | LP RDY | LA38 DECwriter IV, |
| pin 1 | Chassis GND | LA34 DECwriter IV |
| pin 7 | Signal GND | |
| Modem (male) | | |
| pin 2 | Modem RCV Data | |
| pin 3 | Modem Xmit Data | |
| pin 12 | SEC Carrier Detect | |
| pin 5 | PRI CTS | |
| pin 22 | Ring Indicator | |
| pin 8 | Carrier Detect | Bell System type: |
| pin 6 | Data Set RDY | 103, 113, 202, 212 |
| pin 11 | SEC RTS | or equivalent |
| pin 4 | PRI RTS | |
| pin 20 | DTR | |
| pin 17 | SYN CLK R | |
| pin 15 | SYN CLK T | |
| pin 1 | Chassis GND | |
| pin 7 | Signal GND | |
| Terminal 1,2,3 (female) | | |
| pin 2 | Term 1 Xmit Data | LA36 DECwriter II, |
| pin 3 | Term 1 RCV Data | VT50 DECscope, |
| pin 20 | Term 1 RDY | VT52 DECscope, |
| pin 7 | Signal IN (GND) | LA34 DECwriter IV, |
| pin 1 | Chassis IN (GND) | LA38 DECwriter IV, |
| | | VT100 video terminal |
| | | VT105 graphic terminal |

**NOTE**

A typical serial ASCII protocol includes the signals DSR (Data Set Ready), DTR (Data Terminal Ready), CTS (Clear to Send), and RTS (Request to send). Please note the following exceptions for MiniMINC:

1. MiniMINC requires DTR but does not supply DSR.

2. MiniMINC neither requires nor provides CTS or RTS.

# CHAPTER 2
# DATA PROCESSING ROUTINES

## FFT: PERFORM FAST FOURIER TRANSFORM

### Operation

FFT performs a discrete Fourier transform on a data array. The fast Fourier transform algorithm provides an efficient method for numerically approximating a continuous Fourier transform.

The continuous Fourier transform converts, for example, functions in the time domain to functions in the frequency domain. Although the FFT routine is based on the discrete Fourier transform algorithm, it takes advantage of certain computational shortcuts to reduce the time required to produce the results.

FFT calcluates the real and imaginary parts of the Fourier coefficients.

### Statement Form

FFT( ,data-length,real-component,imag-component, scale-factor)

### Argument Descriptions

**option** The character string designating the direction of the transform. Valid values are FORWARD and REVERSE.

FORWARD performs a forward transform.

REVERSE performs a reverse, or "inverse," transform.

If you omit the option word, FORWARD is used by default.

Provided the appropriate scale factors are applied to the results, a forward transform followed by a reverse transform produces results identical to the input to the forward transform.

After a reverse transform, you must divide each element of the resulting arrays by the data-length argument.

**data-length** The number of points in each of the component arrays that are used as input to FFT. Valid values are powers of 2 that range from 8 to 2048. The real-component and imag-component arrays must each contain at least this number of elements. (The actual maximum array length depends on the amount of available workspace; see the LENGTH command in Book 3.)

**real-component** The name of the integer array containing the real component of the (complex) input data that is to be transformed. FFT replaces the input data in this array with the calculated frequency coefficients.

The nature of the FFT algorithm requires that the data satisfy certain conditions. See "Restrictions" for the FFT routine.

**imag-component** The name of the integer array containing the imaginary components of the (complex) input data to be transformed. FFT replaces the input data with the calculated frequency coefficients.

The nature of the FFT algorithm requires that the data satisfy certain conditions. See "Restrictions" for the FFT routine.

If the input data are real, then all elements of the imag-component array should be 0.

**scale-factor** The integer scale factor used to scale the real and imaginary components of the result to provide the final result.

For efficient calculations, FFT requires integer input arrays. However, because integer values are limited in range, FFT requires a mechanism for keeping the data values within the integer range during its calculations. It does this by dividing all the values by 2 whenever necessary to keep within the range. Therefore, the results are proportionally correct but need to be scaled in order to restore the original absolute values. After FFT has finished, the scale-factor argument contains the number of divisions. Create final results in two real arrays by multiplying each element of the integer result arrays by 2 to the power n (where n is the scale factor).

**Related Routines**

1. POWER calculates the power spectrum of a set of data by using the real and imaginary coefficient arrays calculated

by FFT.

**Restrictions**

1. The discrete Fourier transform algorithm approximates the theoretically desired continuous transform. The closeness of the approximation depends on the extent to which the input data satisfy the following conditions:

   - The function to be transformed must be periodic.
   - The function to be transformed must be band-limited. That is, the highest frequency component of the function must be finite.
   - When you collect the input data, the sampling rate must be higher than twice the highest frequency component of the function.
   - The number of input data samples must be an exact integer multiple of the period of the input waveform.

   When the above conditions are not met, discrepancies begin to accumulate between the continuous Fourier transform and its discrete, numerical approximation.

   If the function is not periodic and band-limited, then, by definition, the other conditions cannot be satisfied.

   If the number of samples is not an exact integer multiple of the period of the input waveform, then the FFT results are distorted by *leakage.*

   If the sampling rate is too low, the FFT results exhibit *aliasing.* That is, the FFT coefficients identify frequency components that were not present in the original waveform. Figure 3 illustrates the meaning of aliasing.

   It is important to recognize causes of discrepancy between the discrete Fourier transform and the continuous transform in order to use the FFT routine effectively. Consult a good general text for a complete, formal description of the discrete transform iteself, the discrepancies that can occur, and the methods necessary to minimize the discrepancies. For example, see *The Fast Fourier Transform* by E. Oran Brigham (New York: Prentice-Hall, 1974). See also the manual for Digital's Laboratory Subroutine Package (Digital Equipment Corporation, Order Number AA-C984A-TC).

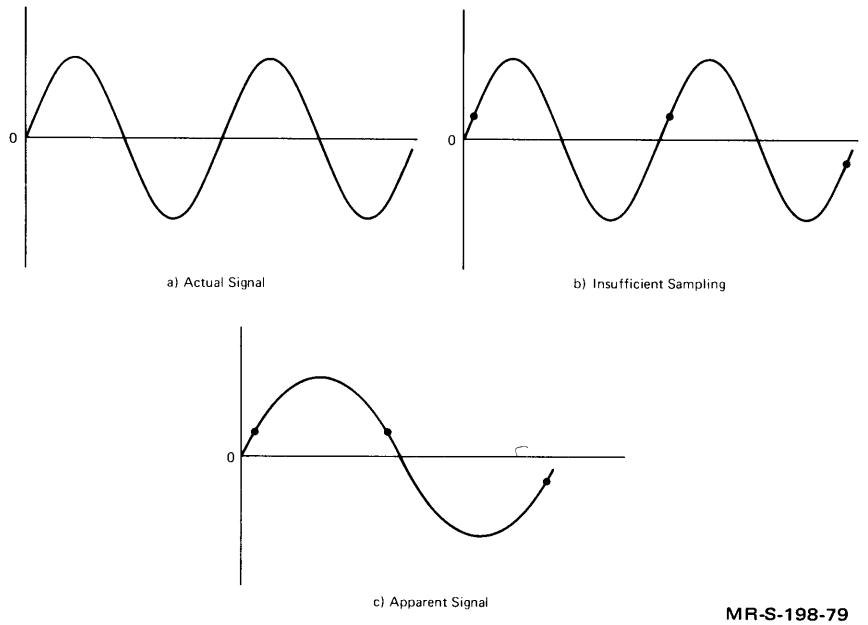



Figure 3. How Aliasing Occurs

2. Two different scale factors affect FFT results.

   If you want absolute values (rather than proportional values), multiply all elements of the real-component and imag-component arrays by 2 to the power n (where n is the scale-factor argument).

   After a reverse transform, you must *divide* each element of the real-component and imag-component arrays by data-length.

**Errors**

?MINC-F-Both REAL and IMAG must be at least as large as SIZE
?MINC-F-SIZE is less than or equal to 8
The number of points requested (SIZE) conflicts with the size of the real or imaginary array, or the number of points requested is less than or equal to 8.

?MINC-F-Invalid or conflicting options requested
FFT permits only a single option word in a statement. If you include an option word, the only valid choices are FORWARD and REVERSE.

?MINC-F-SIZE is greater than 2048
The number of points requested (SIZE) must be a power of 2 that is less than or equal to 2048.



?MINC-F-SIZE is not a power of 2

The number of points requested (SIZE) must be a power of 2 from 8 to 2048.

**Example**

See the program F6FFT.BAS on the demonstration diskette. F6FFT calculates and displays a function that meets the stated conditions for data input to the FFT routine. The program then calculates and displays the 'FORWARD' FFT, the power spectrum, and the 'REVERSE' FFT for the supplied function.

## POWER: CALCULATE POWER SPECTRUM COEFFICIENTS

POWER calculates the power spectrum for a set of complex Fourier coefficients. The power spectrum is the relationship between power level and signal frequency.

**Operation**

POWER is not limited to power spectra for FFT coefficients. You can use POWER to calculate the sums of the squares of any pairs of values.

**Statement Form**

POWER(data-length,real-component,imag-component, spectrum-array)

**Argument Descriptions**

**data-length** The number of elements in each of the three arrays. Valid values are integers equal to or greater than 1.

The destination array for the power spectrum must contain at least data-length elements. In practice, the length of the arrays is limited by the amount of available workspace (see Book 3, LENGTH).

You cannot omit the data-length argument from a POWER statement.

**real-component** The integer array containing one set of coefficients. Valid values are the range of integers.

The real-component array can contain any set of coefficients. One such set of coefficients is the set of real component coefficients produced by the FFT routine.

You cannot omit the real-component argument from a POWER statement.

**imag-component** The integer array containing one set of coefficients. Valid values are the range of integers. The imag-component array can contain any set of coefficients. One such set of coefficients is the set of imaginary coefficients produced by the FFT routine.

You cannot omit the imag-component argument from a POWER statement.

**spectrum-array** The real array to contain the power spectrum coefficients. Valid values are the range of real numbers.

POWER assigns real values to the spectrum-array. You cannot omit the spectrum-array argument from a POWER statement.

**Related Routines**

1. FFT calculates complex Fourier coefficients given a set of complex data. FFT uses a discrete fast Fourier transform algorithm.

**Restrictions**

1. Each power spectrum coefficient is the sum of the squares of the corresponding real and imaginary coefficients.

   The traditional definition of power is the square root of the sum of the squares of the coefficients.

   ```
   P(a)=SQRT(R(a)**2+I(a)**2)
   ```

   Therefore, to obtain actual power coefficients, take the square root of each element in the spectrum array.

   In general, POWER and FFT are used to obtain proportionally correct values, not absolute values.

**Errors**

?MINC-F-SIZE must be greater than or equal to REAL, IMAG, & PSPECT arrays.

You must request a number of points greater than or equal to the real, imaginary, and spectrum arrays.

?MINC-F-SIZE must be positive
Number of points requested (SIZE) must be a positive integer.

**Example**

See the program F6FFT.BAS on the demonstration diskette.

# CHAPTER 3
# PROGRAM CONTROL ROUTINES

## SCHEDULE: SCHEDULE PROGRAM RESPONSE TO A TIME EVENT

**Operation**

SCHEDULE designates a service subroutine and schedules it to execute when a specified time event occurs. The time event is either the completion of a specified time interval or the occurrence of a specified time of day. The program continues executing until the time event occurs.

**Statement Form**

SCHEDULE(option,time,subroutine)

**Argument Descriptions**

**option** The character string specifying whether the time event is a time interval or an absolute time of day. Valid values are ABSOLUTE, INTERVAL, and no option. If you do not include an option, INTERVAL is used by default.

The longest time interval possible with either mode is 24 hours. In interval mode, you can schedule a time event for 24 hours (86,400 seconds) later. In absolute mode, if the time string represents a time earlier than the current time, SCHEDULE schedules the time event for the next day.

Absolute time of day is expressed using the 24-hour format, that is, time measured relative to midnight. Five o'clock in the morning is 5 hours past midnight; five o'clock in the afternoon, 17 hours.

**time** The time of day or time interval required. Valid values for time of day are the *strings* 0 to 24:00:00, giving the time in hours, minutes, and seconds past midnight. Valid values for time intervals are real numbers from 0 to 86,400, giving the time interval in seconds. You can also use the string format to specify time intervals.

If you omit the time argument, 0 (or 0:00:00) is used by default.

When the time argument is a numeric expression, 0.1 is the shortest interval guaranteed to be precise. SCHEDULE does not consider times shorter than 0.1 to be errors, but these times are not precise. If the time argument is 0, SCHEDULE schedules the time event for the next tick of the system clock.

When the time argument is a string expression, the string has the frame "hours:minutes:seconds". Each of the slots (hours, minutes, seconds) can hold up to two digits or can be empty. The largest time you can specify is 24:00:00. The string itself can contain only digits and colons; blanks are invalid.

If the string contains only one number, SCHEDULE assumes that the number specifies seconds, unless the number is followed by one or two colons. The colons indicate the position of the number in the string frame. For example, the string '8' means 8 seconds, the string '8:' means 8 minutes, and the string '8::' means 8 hours. The string ':8:' is equivalent to '8:'.

**subroutine** The statement number of the service subroutine to execute when the time event occurs. Valid values are 0 and the range of integers from 1 to 32767.

A value of 0 means "cancel the scheduled request."

Values from 1 to 32767 give the line number of the service subroutine.

If you omit the subroutine argument, the value 0 (cancel request) is used by default.

MiniMINC transfers program control to the specified statement at the end of the time interval or at the specified time of day.

**Related Routines**

1. PAUSE suspends program execution for a specified time interval. After the time interval expires, program execution continues with the statement following the PAUSE statement.

**Restrictions**

1. A scheduled routine will not be executed until a current graphic operation terminates.

2. With absolute time specifications, if the current time is later than the time requested, the event is scheduled for the next day. As a result, no event can be scheduled more than 24

hours in advance.

3. A SCHEDULE statement with the subroutine argument omitted cancels a pending request. The cancellation request executes without error (but does nothing) if no request is pending.

4. The SCHEDULE routine does not operate in immediate mode.

5. To replace a pending request, you cancel the first request and then schedule a new request.

6. The resequencing command (RESEQ) changes statement numbers in a program. However, RESEQ does not change the subroutine argument in a SCHEDULE statement. When you resequence a program, you must determine the new statement number of the service subroutine and then change the SCHEDULE statement's subroutine argument accordingly.

   For convenience, we suggest that you use a variable name for the subroutine argument. Put the variable assignment statement at the beginning of the program, with an explanatory remark. Then, when you resequence the program, you can quickly change the assignment statement, and you will not have to search for each occurrence of the SCHEDULE statement (unless different SCHEDULE statements refer to different service subroutines).

7. SCHEDULE schedules only one time event at a time. When the requested time event occurs, the request has been satisfied and there is no longer any request pending. Therefore, to schedule repetitive events, use another SCHEDULE statement in the service subroutine, as in

```
100 SCHEDULE("ABSOLUTE","11:30:15",500)

.
.
.
500 REM Service subroutine begins

[Statements of subroutine]

.
.
.
```

```
550 SCHEDULE("ABSOLUTE","13:30:15",500)
560 REM End of service subroutine
570 RETURN
```

8. You can change the system time with the TIME command (see Book 3, TIME). If the system time you set is not the actual time of day, time intervals are still measured correctly, but absolute times are not.

9. While the program is waiting as the result of a PAUSE statement, it does not perform any scheduled service subroutines. Therefore, PAUSE should not be placed after a SCHEDULE statement in an attempt to wait for the pending service request.

**Errors**

?MINC-F-Could not find service subroutine #### requested
You specified an invalid line number for the service subroutine.

?MINC-F-Invalid or conflicting options requested
The option you specified was invalid. If you include an option word, INTERVAL and ABSOLUTE are the only valid choices.

?MINC-F-Previously scheduled event pending. No new request.
Only one request can be active at a time. You cannot schedule multiple time events.

?MINC-F-Time string format must be hh:mm:ss
Reenter time string in correct format.

?MINC-F-Value of argument 2 is outside valid range
The longest valid time interval is 86,400 seconds.

?MINC-F-Data transfer or pending service request terminated
The scheduled action could not begin executing, because some other process was terminated. This error may occur if a program finishes without executing a service request or if CTRL/C is used prematurely.

**Examples**

None.

## PAUSE: SUSPEND PROGRAM EXECUTION

**Operation**

PAUSE suspends program execution for a specified time interval. After the time interval expires, program execution continues with the statement following the PAUSE statement.

**Statement Form**

PAUSE(delay-interval)

**Argument Descriptions**

**delay-interval** The length of time to wait before continuing execution. You can express the interval as a real number indicating the number of seconds to wait; valid values are 0 to 86,400. You can also express the interval as a string in the format "hours:minutes:seconds"; valid values for strings are 0:00:01 to 24:00:00.

You cannot omit the delay-interval argument.

When the delay-interval argument is a numeric expression, .1 second is the shortest time interval guaranteed to be precise. Shorter intervals are accurate to the nearest clock tick.

When the delay interval is a string expression, one second is the shortest possible interval. The time string has the frame "hours:minutes:seconds". Each of the slots in the frame can hold up to two digits or can be empty. The maximum interval possible in the time string is 24:00:00. The string itself can contain only digits and colons; blanks are invalid.

If the string contains only one number, PAUSE assumes that the number specifies seconds, unless the number is followed by one or two colons. The colons indicate the position of the number in the string frame. For example, the string '5' means 5 seconds, the string '5:' means 5 minutes, and the string '5::' means 5 hours. The string ':10:' is equivalent to '0:10:0' and also to '10:'.

**Related Routines**

1. SCHEDULE designates a service subroutine and schedules it to execute after a specified time interval or at an absolute time of day. The program continues executing during the time interval.

**Restrictions**

1. The CTRL/C key combination works normally during a pause. In fact, CTRL/C is the only method available for aborting a pause. We advise you not to use the RCTRLC function (see Book 3) to disable CTRL/C during pauses.

2. PAUSE accepts long time intervals. Therefore, if you initiate a long pause, the system might appear broken or inactive. Some other user may mistakenly enter CTRL/C, terminating your program. To avoid such problems, print an advisory message on the screen, such as:

```
110 PRINT T$' hour delay in progress, started at 'CLK$
120 PAUSE(T$)
```

or

```
110 PRINT T' second delay in progress was started at ';CLK$
120 PAUSE(T)
```

With valid long delays that span midnight, MiniMINC correctly changes the time from 24:00 to 0:00.

3. PAUSE accepts only time interval arguments. As a result, you cannot specify an absolute time of day as the end of a pause. There are two solutions to this restriction.

   You can compute the time interval between the current time and the desired time and use that interval in a PAUSE statement.

   You can use SCHEDULE with the desired time of day as the argument and follow the SCHEDULE statement with a loop that continues to execute until the time of day arrives. For example:

```
50 F=0
60 SCHEDULE('Absolute','12::',200)
70 IF F=0 GOTO 70
.
.
.
199 STOP
200 REM Service subroutine for SCHEDULE
210 F=1
220 RETURN
.
.
.
```

4. While the program is waiting as the result of a PAUSE statement, it does not perform any scheduled service subroutines. Therefore, PAUSE should not be placed after a SCHEDULE statement in an attempt to wait for the pending service request.

**Errors**

?MINC-F-Time string format must be hh:mm:ss
If you enter a time string instead of a real argument, the time string must be in the form "hh:mm:ss", such as "13:38:17".

?MINC-F-Value of argument 1 is outside valid range
You entered a time argument longer than 24 hours. Note that delay intervals shorter than 1 second may not generate an error message, but will result in a delay of 1 second.

?MINC-F-Data transfer or pending service request terminated
This message is produced when you terminate a PAUSE action by a CTRL/C command.

## Examples

None.

# CHAPTER 4
# GRAPHIC ROUTINES

## FIND_CURSOR: CHANGE AND RECORD CURSOR POSITION

### Operation

A FIND_CURSOR statement allows you to move the cursor to any position on the screen, using the left-, right-, up-, and down-arrow keys on the terminal keyboard.

The execution of a FIND_CURSOR statement in a program causes the program to pause, at which point the cursor can be moved with the arrow keys. When you have the cursor at the desired position, strike any alphanumeric key (a-z, A-Z, 0-9, or <RETURN>) to terminate the FIND_CURSOR operation. When you strike the alphanumeric key, FIND_CURSOR records the position of the cursor (by row and column), and optionally, the terminating character you typed.

If you enter an invalid terminating character, such as '+', the keyboard beeps. The invalid character is rejected, and FIND_CURSOR continues waiting for a valid terminating character or for further strokes of the arrow keys.

FIND_CURSOR allows you to specify the initial position of the cursor. If no valid position is specified, the cursor begins at row 11, column 40.

FIND_CURSOR provides a convenient means of displaying interactive "menus" in an application program. For an example, see the example section for this routine.

### Statement Form

FIND_CURSOR(ROW,COLUMN,CHARACTER)

### Argument Descriptions

**ROW** The name of the variable that will store the row number. Any valid numeric variable name can be used, but we suggest that, for maximum clarity, you use an integer variable with a

mnemonic name, such as R%. If, before the execution of the FIND_CURSOR statement, ROW has been assigned the value of a valid row number (1-24), then the cursor will appear initially at that row. A value can be assigned either by a previous FIND_CURSOR statement or by a previous assignment statement.

ROW is a required argument.

**COLUMN** The name of a variable that will store the column number. Again, we suggest a name such as C%. If, before the execution of a FIND_CURSOR statement, COLUMN has been assigned the value of a valid column number (1-80, 1-132), then the cursor will appear initially at that column. A value can be assigned either by a previous FIND_CURSOR statement or by a previous assignment statement.

COLUMN is a required argument.

**CHARACTER** The name of the string variable that will store the terminating character. Do not use a numeric variable. We suggest a mnemonic name such as C$.

CHARACTER is an optional argument. If you omit CHARACTER from a FIND_CURSOR statement, FIND_CURSOR still returns the final row and column position of the cursor, but does not return the terminating character. You still must type a terminating character when you omit the CHARACTER argument.

**Related Routines**

1. MOVE_CURSOR allows a program to move the cursor to a given row-column position. However, MOVE_CURSOR requires that you supply exact row-column numbers as arguments.

2. GET_CURSOR also acquires the current location of the cursor. However, GET_CURSOR does not allow moving the cursor as part of the same operation.

3. If you use FIND_CURSOR to move the cursor, it is straightforward to display a text string or symbol at the final cursor position, with the routines HTEXT, VTEXT, and PUT_SYMBOL, respectively. It is also possible to use the terminating character to indicate the character mode for the displayed string or symbol. See "Examples" for this routine.

4. In "menu" programs, the variables used for FIND_CURSOR arguments can be used to control the execution of other statements in the program. Because it is a string expression, the value of the CHARACTER argument can be used to select an option word in routines that use option words. See "Examples" for this routine.

5. The statement DISPLAY_MODE("LONG") changes the range of valid column numbers from 1-80 to 1-132.

6. The FIND_POINT routine is the graphic analog to FIND_CURSOR. FIND_POINT allows movement of a graphic "cursor" (a brand) to points within a graphic region.

**Restrictions**

1. In a series of FIND_CURSOR statements, the cursor can begin where it was left by the previous FIND_CURSOR statement. To use this feature, however, you must use the same variable names for arguments in every FIND_CURSOR statement in the series.

2. If you put a FIND_CURSOR statement in a program loop, the terminating characters may be echoed on the screen.

**Error**

?MINC-F-Variable name required for argument x
You included an argument that was not a valid variable name. Repeat the statement with valid names.

**Examples**

```
FIND_CURSOR(R,C,M$)
PRINT 'Text to display'; \ LINPUT T$
HTEXT(M$,R,C,T$)
```

allows you to move the cursor to a desired location with the arrow keys. When the cursor is in place, strike the letter B, R, U, or F. The text string you supply to the INPUT statement will be displayed at the cursor location in boldface, reverse video, underlined, or flashing characters, respectively.

The program MENU, supplied on your demonstration diskette, shows FIND_CURSOR being used to select program statements from a menu. The program uses the returned row number to identify a particular statement for execution. The terminating character is used for an option word in some of the selected statements.

To run the program, put a system diskette in SY0: and a demonstration diskette in SY1:. Then type RUN SY1:MENU.

# CHAPTER 5
# MISCELLANEOUS ROUTINES

## GET_CHAR: ACQUIRE SINGLE CHARACTER FROM TERMINAL

### Operation

This routine acquires a single character from the keyboard. If used in a program, GET_CHAR does NOT make the program pause and wait for input (unlike the INPUT statement of BASIC). Instead, it returns a character only if one was received since the last time the program retrieved input.

If a character was previously typed, GET_CHAR stores it in a specified string variable. If no character was typed, this variable will be a "null" (empty) string.

In certain application programs, GET_CHAR can be useful because it allows you to interrogate the operator of the program without requiring that he respond at any particular point in the program's execution. An example follows at the end of this section.

### Statement Form

GET_CHAR(CHARACTER)

### Argument Descriptions

**CHARACTER** The name of a string variable. We suggest a mnemonic name such as C$. This variable will be set by GET_CHAR either to the null string or to the character detected by the routine. CHARACTER cannot be omitted from the statement.

### Related Routines

1. SYS(1) takes on the numeric code of the next ASCII character received by the terminal. However, you must type an entire line terminated by a (RET) before even the first character is available to SYS(1).

**Restrictions**

1. Unlike the FIND_CURSOR routine, GET_CHAR echoes any character you type. Some characters may not be echoed if the GET_CHAR statement is in a tight loop.

2. GET_CHAR properly retrieves the character SPACE (ASCII 32). However, comparison statements in MINC BASIC generally fail to distinguish SPACE from NULL (ASCII 0).

   Consider:

```
10 GET_CHAR (C$)\REM Get a character
20 IF C$><CHR$(0) THEN 100
30 GO TO 10
.
.
.
100 REM Process received character
```

In this sequence, C$ will be a SPACE if one was typed, but the program will not branch to line 100, because line 20 will fail to distinguish SPACE from NULL.

**Error**

```
?MINC-F-Variable name required for argument x
```

The argument you entered was not a valid string variable name. Repeat the statement with a valid name.

**Example**

```
10 REM DEMONSTRATION OF GET_CHAR ROUTINE
20 REM ********************************
30 FOR I=0 TO 10000
40 GET_CHAR(C$)
50 IF C$='S' THEN 100
60 PRINT I,SIN(I*PI/50)
70 NEXT I
80 REM ********************************
90 REM GOES TO LINE 100 IF CHARACTER 'S' IS TYPED
100 STOP
```

This program prints 10001 sine values unless interrupted by a stroke of the S key. Any character you type while the program is running is received by the GET_CHAR statement on line 40. The IF statement on line 50 checks to see whether an S was typed. If S was typed, the program branches to line 100 and stops; otherwise, the program keeps printing I and the sine expression.

# CHAPTER 6
# CHANGING OPERATING MODES ON THE MiniMINC TERMINAL (SETUP)

The MiniMINC terminal is a member of a new generation of computer-related devices that not only give their users a large number of useful operating characteristics but also have a significant amount of built-in intelligence and memory. The latter two attributes permit you to choose operating characteristics by keyboard command rather than by setting hardware switches. Selecting these characteristics is accomplished in what is called setup mode.

## WHEN TO USE SETUP MODE

Enter setup mode whenever you become aware that the terminal is set to a parameter that is inconsistent with your immediate objectives — when you want to change the background shade, for example, or increase or decrease screen contrast. You may have to change the setup parameters when a previous user has abandoned the system without restoring the terminal to its normal operating state. Checking the setup parameters is a routine step in troubleshooting the system (see "TROUBLESHOOTING," below).

You should have access to a MiniMINC terminal while reading the following section so that you can execute the commands in question and get first-hand evidence of their effect.

## HOW TO USE SETUP MODE

You can enter setup mode at any time by pressing the SETUP key at the upper left corner of the MiniMINC keyboard. This causes the Setup A display to appear on the screen, as shown in Figure 4. At this point, all transmissions from MiniMINC to the

terminal are suspended until you strike the SETUP key again. There is also a secondary display, Setup B, which is discussed later in this section.

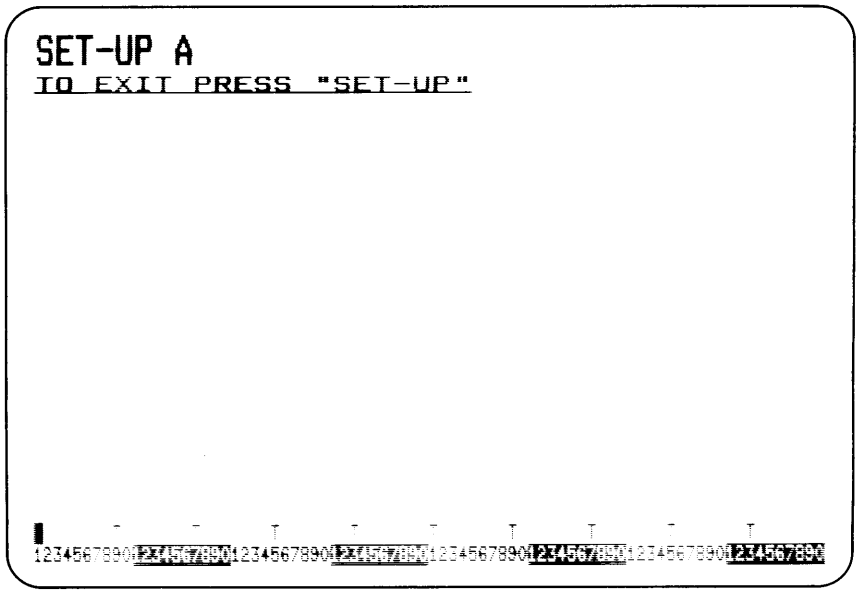



Figure 4. Setup A Display MR-S-199-79

You can leave setup mode at any time by pressing the SETUP key a second time. For the present, leave the setup A display on the screen.

**NOTE**

Entering setup mode does not erase your text from the character storage memory in the terminal but only replaces it temporarily on the screen with the setup displays. (Note that any graphics display details remain on the screen with the setup information.) When you press the SETUP key a second time to leave setup mode, your former text is restored. The only setup mode keys that DO erase the terminal character storage memory are the 9 key, the 0 key, and the SHIFT/R key. These keys are described in this section.

**Setup A**

* Screen contrast. The contrast of the screen is increased by the up-arrow key (↑) and decreased by the down-arrow key (↓). Hold down each of these keys and notice the change in the screen contrast.

* Tab stops.

**NOTE**

Tab stop settings on the terminal are ignored by the MiniMINC system. The tab stop setting procedure is given here for completeness, but you should not expect these settings to affect text output by your MiniMINC programs.

You can change the tab stop setting by pressing the 2 key. Tab stops are shown on the screen by the letter T, which appears at various points along the list of numbers (the Ruler) at the bottom of the Setup A display. This Ruler defines the number of character columns on the screen. To change a tab stop, move the cursor to the position above the desired column and then press the 2 key. (The left- and right-arrow keys, the tab key, the return key, and the space bar all move the cursor without affecting setup conditions.) If there is already a tab stop at the column when you press the 2 key, that stop is cleared. If there is no previous tab in the column, a T appears, meaning that a tab stop is set at that position. You can repeat this operation as many times as there are character columns on the screen.

* Tab clear. See note under "Tab stops," above. To clear all tab stops, press the 3 key.

* Local mode. The MiniMINC terminal can operate in two modes, local and remote. In the former, the screen image can be modified by operating the keyboard, but no data are transmitted to or received from the MiniMINC system. In remote mode, anything displayed on the screen (even the characters that you type) is received from a remote source — in this case, the MiniMINC system. Since a terminal that will operate properly in local mode will normally operate properly in remote, local mode is useful for confirming that display problems are not the fault of the terminal.

To put the terminal in local mode, press the 4 key. This action will turn on the LOCAL lamp at the top of the keyboard. If you leave setup mode with the terminal set to local, the characters you type will appear on the screen but will not be sent to the MiniMINC system.

The normal MiniMINC terminal operating mode — allowing standard communication with the MiniMINC system — is re-

mote. To return to remote mode, press the 4 key again. This will cause the on line lamp to turn on.

* Screen width. The 9 key controls the width of the screen in columns. Press this key and observe that the ruler along the bottom of the screen changes. The screen is now 132 columns wide. Press the key again and the ruler reverts to its normal width of 80 columns. THE NORMAL 80-COLUMN WIDTH IS EXPECTED BY MiniMINC GRAPHIC ROUTINES. The graphic routine DISPLAY_MODE allows you to select screen widths of 80 or 132 characters; use DISPLAY_MODE rather than setup mode for this purpose.

**NOTE**

Character data is lost from the character storage area when the 9 key is pressed.

* Terminal test. The 0 key starts a test procedure inside the MiniMINC terminal which checks the keyboard and portions of the terminal's memory. Pressing this key generates a rapid series of character displays on the screen. If the test is successful, it ends by restoring the permanent setup mode parameters (see the next item) and exiting from setup mode.

If the test fails, error information is generated on the keyboard lamps and the terminal screen. See "Troubleshooting MiniMINC" for more information.

**NOTE**

Pressing the 0 key also erases any text (but not graphics) that was in the terminal character storage memory before you entered setup mode.

* Saving setup conditions. The SHIFT/S key saves in the terminal's control memory whatever terminal conditions are currently specified. These parameters will remain in the control memory even when the terminal is turned off. Saved parameters are restored when the power is turned on again, at the completion of the terminal test, and when you press SHIFT/R in setup mode.

**NOTE**

Be careful with the SHIFT/S key. Do not save setup mode parameters unless you are sure that they will not interfere with the needs of other users of the MiniMINC system. See Figures 4 and 5 for standard MiniMINC parameters.

* Restoring previously saved setup conditions. The SHIFT/R key RESTORES to control memory whatever operating parameters have been previously saved — in the case of a new machine on which no variant parameters have been saved, those shown in Figures 4 and 5.

* Answer back message. SHIFT/A elicits "A=" from the terminal, after which the user of certain non-MiniMINC systems may enter an answer-back message for system use. This feature is not used by the MiniMINC system. If you inadvertently type SHIFT/A, respond to "A=" with a carriage return.

**Setup B**

The remaining features of setup mode are accessed by means of Setup B, which you enter by pressing the 5 key. This causes the screen to display the pattern shown in Figure 5. You can restore the Setup A display by pressing the 5 key a second time — but leave the Setup B display on the screen while you read the following.

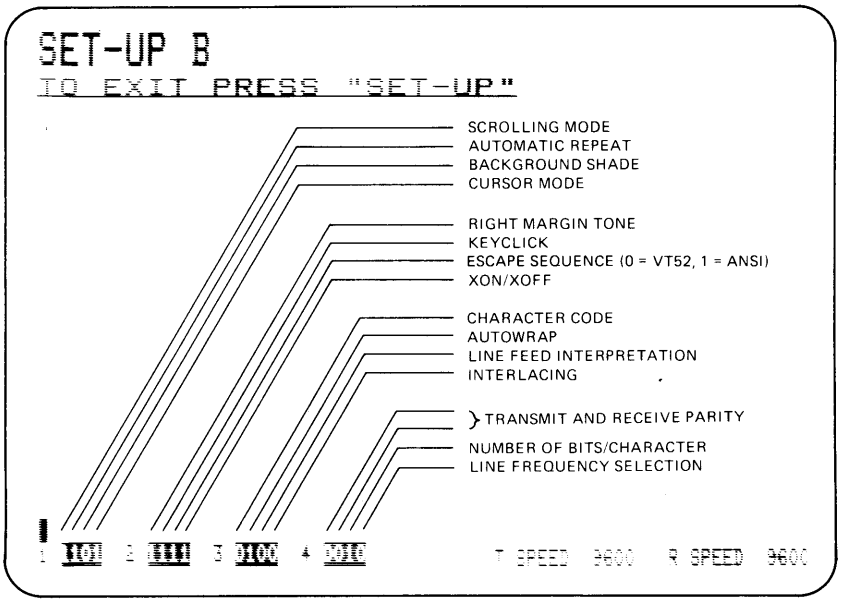



Figure 5. Setup B Display

MR-S-200-79

**NOTE**

If you wish only to verify that your terminal is properly set up for MiniMINC programming, compare the Setup B display on your screen with Figure 5, which shows a set of values that is compatible with all MiniMINC routines.

If your screen reveals a different setup from that indicated in Figure 5, press SHIFT/R and then press the 5 key again. If your setup still differs from that defined in Figure 5 in some positions, reset these positions according to the instructions in this section.

Notice that, in Setup B, the cursor is located at the left end of the groups of 0's and 1's at the bottom of the screen. Each of these digits represents what amounts to an on-off switch, with 0 signifying *off* and 1 signifying *on*. To control the remaining setup mode parameters, move the cursor with the left arrow (←) or right-arrow (→) keys until it is located over a digit you want to change; then press the 6 key. This action changes the number under the cursor either from 0 to 1 or from 1 to 0. Numbering of groups and digits within groups is from left to right and begins with 1.

The effect of each control digit in Setup B is discussed below. Control digits are identified for quick reference in Figure 5.

* Scrolling mode. Digit 1 of group 1 controls scrolling mode. If the scrolling digit is 0, the terminal scrolls by jumping each line of characters up one line. If the scrolling digit is 1, the terminal scrolls with a smooth continuous movement of all affected lines. Smooth scrolling displays characters at a slower rate than "jump" scrolling.

Set this digit to 1 on all MiniMINC systems.

The graphic routine DISPLAY_MODE allows you to select either smooth or jump scrolling; use DISPLAY_MODE rather than setup mode for this purpose.

* Automatic repeat. Digit 2 of group 1 controls the automatic repeat feature for the terminal keys. If this digit is 0, none of the keys repeat. If this digit is 1, most keys will repeat automatically if you hold them down instead of depressing them momentarily. The keys that do not repeat in either case are SETUP, ESC, NO SCROLL, TAB, RETURN, and CTRL. You may choose the set-

ting that you find most convenient.

* Background shade of screen. Digit 3 of group 1 controls the figure-ground relationship of characters on the screen. If this digit is 1, the screen displays black characters on a white background. If the digit is 0 (the normal setting), the terminal displays white characters on a black background. MiniMINC GRAPHIC ROUTINES INITIALLY EXPECT BLACK BACKGROUND SHADING.

* Cursor mode. Digit 4 of group 1 controls cursor mode. When this digit is 0, the cursor appears as a flashing underscore character. When this digit is 1, the cursor is a flashing white square on a black ground or vice versa. You can use either setting with MiniMINC.

* Right margin tone. Digit 1 of group 2 controls the right margin tone. When this digit is 1, a tone sounds when your typed line comes within eight columns of the right margin. The tone does not sound if this digit is 0. You can use either setting with MiniMINC.

* Keyclick. Digit 2 of group 2 controls the "keyclick" feature of the terminal. If the digit is 1, any keystroke on the terminal produces an audible click. This click provides audible feedback for users who are accustomed to typewriters. If the digit is 0, all the keys are silent. You can use either setting with MiniMINC.

* Escape sequence mode. Digit 3 of group 2 MUST be set to 1, to comply with the ANSI standard for escape sequences. THIS SETTING IS REQUIRED BY MiniMINC. (If the digit is 0, the terminal emulates the escape sequences used by Digital's VT52 terminal, and MiniMINC routines will not function properly.)

* Automatic XON/XOFF. Digit 4 of group 2 MUST be set to 1, enabling this feature. REQUIRED FOR ALL MiniMINC OPERATIONS. (With this feature enabled, the processor is automatically commanded to suspend data transmission when the data handling speed of the terminal is exceeded. When the terminal catches up, the processor is automatically commanded to resume data transmission.)

* Character code selection. Digit 1 of group 3 MUST be set to 0, for ASCII code. REQUIRED FOR ALL MiniMINC OPERATIONS. (A setting of 1 selects the United Kingdom code, which is incompatible with the MiniMINC routines.)

* Autowrap feature. Digit 2 of group 3 MUST be set to 1, enabling this feature. (A setting of 1 causes all lines exceeding the width of the screen to "wrap around" to the following line and thus permits displaying lines whose length exceeds the selected screen width.)

* Interpretation of RETURN key. Digit 3 of group 3 MUST be set to 0. REQUIRED FOR ALL MiniMINC OPERATIONS. (The 0 setting creates a single carriage return character in response to a RETURN keystroke. A setting of 1 appends a LINE FEED character as well, causing the RETURN key to be interpeted as a "new line" function; that is, the carriage return moves the cursor back to the left margin of the current line, and the subsequent line feed moves the cursor down to the beginning of the next, or "new" line.) The MINC system appends a line feed character to all carriage return characters, so when digit 3 is set to 0, the net effect is to generate a new line for each stroke of the RETURN key.

* Interlacing. Digit 4 of group 3 controls the number of scan lines on the screen. If you look closely at characters on the screen, you can see the scan lines running through them. If the interlacing digit is 0, there are 240 scan lines between the top and bottom of the screen. If this digit is 1, an additional 240 lines are interlaced between the normal lines. Interlacing makes characters look smoother, since more lines are used to form characters of the same size. Interlacing may cause a slight apparent jitter in the screen image; this is normal and should be no source of concern. The increased resolution that interlacing provides is useful when photographs of the screen are being made since it reduces the amount of undefined space within each character and thereby increases character definition. You can use either setting with MiniMINC.

* Transmission and reception parity. Set both digits 1 and 2 of group 4 to 0 (no parity). REQUIRED FOR ALL MiniMINC OPERATIONS.

* Number of bits per character. Digit 3 of group 4 MUST be set to 1 to indicate 8 bits. REQUIRED FOR ALL MiniMINC OPERATIONS. (A setting of 0 indicates 7 bits.)

* Line frequency selection. Setting digit 4 of group 4 to 0 configures the terminal appropriately for a 60-Hz AC line; setting this bit to 1 configures the terminal for a 50-Hz line. Users in North America should set this digit to 0.

* Transmission speed of terminal (T SPEED). If you hold the 7 key down, T SPEED cycles through a range of baud rates from 50 to 19,200. T SPEED MUST be set to 9600. REQUIRED FOR ALL MiniMINC OPERATIONS.

* Reception speed of terminal (R SPEED). Controlled similarly to T SPEED, but with the 8 key. R SPEED MUST be set to 9600. REQUIRED FOR ALL MiniMINC OPERATIONS.

Now return your terminal to normal operation by pressing the SETUP key again. Be sure that you have left all the setup mode parameters in their required states (see Figures 4 and 5).

# CHAPTER 7
# TROUBLESHOOTING

*Trouble*, for a MiniMINC user, is any system response that violates that user's expectation about how the system should behave. Much more often than not, trouble occurs because you have issued incompatible or inappropriate commands to the system. Assistance in identifying such errors is provided in Book 2, Chapter 16. Make sure you fully understand each command and routine that you use. Pay particular attention to the Restrictions section associated with the discussion of each routine in Books 3 and 4.

Less likely for MiniMINC users, but not to be overlooked, is trouble caused by improper hardware configuration. If the MiniMINC system is to operate correctly, all system interconnections must be appropriately and completely made, power switches must be on, and the system terminal must be set up with the proper operating parameters (see "Changing Operating Modes on the MiniMINC Terminal," above).

MiniMINC systems are fabricated from high quality computer grade components, and all electrical subassemblies are operated under power for an extended period prior to final calibration and testing. This screens out the vast majority of potential failures before the system ever leaves the factory. However, no fault prevention procedure can be perfect, and, over a period of time, faults may develop. For that reason, testing and diagnosing aids are provided with the MiniMINC system. These are discussed below.

1. VT105 self-test. The Setup A 0 command causes the terminal to check the keyboard and its own memory (see

"Changing Operating Modes on the MiniMINC Terminal"). Errors are indicated by the keyboard lamps L1-L4 and/or the presence on the screen of any character other than the cursor after completion of the self-test. If an error is indicated, record the exact lamp condition and screen condition and notify Digital. (Check first to be sure that the keyboard is properly connected to the terminal.)

The VT105 self-test also executes automatically whenever a TEXT_INIT statement is executed.

2. Self-test. Each MiniMINC system is equipped with a diagnostic program in read-only memory that causes the system to exercise and evaluate most of the internal components each time the system is powered up. This program permits, at the user's discretion, exercising the disk drives and the I/O ports.

**NOTE**

More extensive testing is available for the MiniMINC internal components. See Appendix A for information on using the PDT-11/150 System Exerciser program.

## TROUBLESHOOTING PROCEDURE

The following list enumerates the more common user-correctable sources of trouble with the MiniMINC system. These should be checked first when the system fails to operate or operates in what appears to be a strange or sporadic fashion.

1. Main power cable is disconnected or loose.

2. A fuse has blown on the terminal or the MiniMINC chassis. To check for a blown fuse, disconnect the system or terminal power cord, remove the fuse holder and visually inspect the fuse. If a fuse has blown, replace it with a fuse of an *identical rating and type*, reconnect the power cord, and turn the system or terminal on. If the fuse blows again, contact your MINC Customer Support Center (see MiniMINC Release Notes).

3. Signal cables between the system and peripheral equipment are disconnected or loose. See Figure 6 for cable connection information.

4. External apparatus is not operating according to expectation and is monopolizing or overrunning the system.

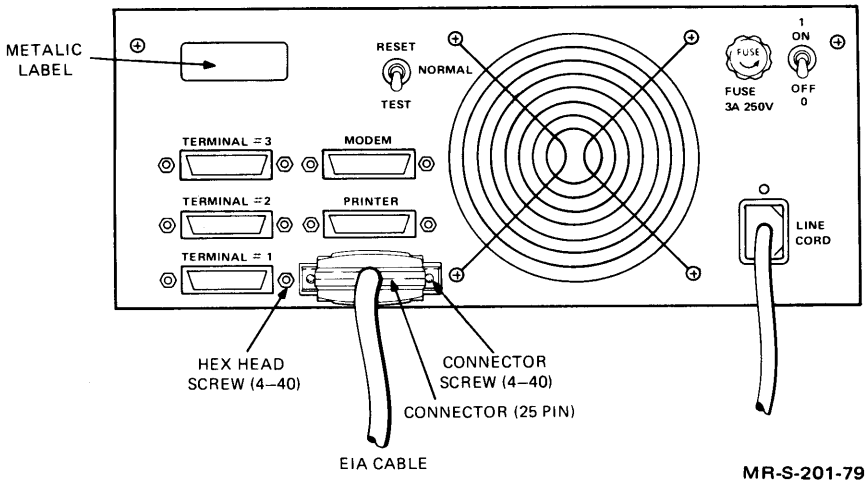



Figure 6. MiniMINC Cable Connections

5. Diskette is improperly inserted.

6. Wrong diskette has been inserted.

7. Bad diskette has been inserted.

8. System terminal has been set up with inappropriate parameters (see "Changing Operating Modes on the MiniMINC Terminal").

## EXECUTING THE MiniMINC SELF-TEST

To run the diagnostic program stored in the MiniMINC read-only memory, proceed as follows (references are to Figure 6):

1. Set the MiniMINC terminal power switch to ON. Wait 30 seconds for the terminal to warm up.

   If the flashing cursor appears in the upper left hand screen area, go to Step 2. If anything other than a flashing cursor appears on the screen, turn the terminal off, then on. If the fault condition persists, make a record of the screen pattern, and contact your MINC Customer Support Center.

2. Confirm that the disk drive doors are up.

3. Set the MiniMINC mode switch to TEST.

4. Set the chassis power switch to ON.

   All chassis lamps except RUN should turn on initially. Then lamps 1 and 2 should blink for 10 to 15 seconds. After

this interval, lamp 1 should turn off and lamp 2 remain on, indicating a successful preliminary test. If this activity does not occur as described, set power switch off, then on. If test fails three times, contact your MINC Customer Support Center.

5. Press "@" ("SHIFT/2") on the keyboard two or three times at half-second intervals. This action allows the MiniMINC chassis to set its baud rate to match that of the terminal.

   Lamp 2 should turn off and the terminal should display:

   ```
   SCRATCH FLOPPY INSTALLED?
   ```

   If this does not occur, set chassis power switch off, then on, and repeat the sequence to this point. If the prescribed events do not occur within three tries, contact your MINC Customer Support Center.

6. Insert an initialized diskette in each drive and close doors.

**WARNING**

This portion of the self-test writes and reads test data over the entire surface of the diskette. DO NOT USE DISKETTES CONTAINING USEFUL PROGRAM OR DATA FILES.

7. Type "Y (RET) " on the keyboard. This affirms your wish to run the disk test. The system will respond with:

   ```
   EIA LOOPBACK TEST?
   ```

   The loopback test requires a 7421963 loopback connector, not normally provided with MiniMINC systems. Type "N (RET) " to signify that you do not want to run the loopback test.

**NOTE**

A significant amount of testing of the serial input/output electronics is performed automatically by the self-test before the loopback test is offered. If executed, the loopback test would add to what has already been done by testing the line drivers and receivers.

8. The system now writes and reads data on selected tracks, first on drive 0, then on drive 1. Allow approximately 2 minutes to complete the disk test.

   If the disk test is successfully completed, the system displays:

   "xxxxxx" (the six digits may have any value)

   In this case, go to step 9.

   If an error is encountered during the disk test, lamp 1 will turn on. In this case, set the chassis power switch off, then on, and repeat entire test procedure. If an error-free pass cannot be achieved in three tries, contact your MINC Customer Support Center.

9. Momentarily set the mode switch to RESET, then allow to return to NORMAL. Lamps 1 and 2 should blink slowly for several seconds, then lamp 2 should remain on. This signifies that the internal processor and memory tests have been successfully executed. If lamp 1 remains on, repeat this step. If an error-free pass cannot be achieved in three tries, contact Digital Field Service.

10. Press "@" ("SHIFT/2") two or three times at half second intervals. This causes the MiniMINC chassis to set itself to a baud rate that matches that of the terminal. Lamp 2 should turn off and the terminal should display:

```
READ ERROR
Type start unit number (0 or 1):
```

    or

```
NO BOOT ON VOLUME
Type start unit number (0 or 1):
```

    Neither of these messages signify an error at this point. After setting its baud rate, the system automatically tries to start itself on the expectation that a diskette with a valid self-starting (bootstrap) routine resides in drive 0. If you have just finished the disk drive test, drive 0 contains a blank diskette and the system so informs you with the "NO BOOT ON VOLUME" message. If no diskette is present, the system issues the "READ ERROR" message and waits for you to insert a system diskette containing an appropri-

ate bootstrap routine. If one of the above messages does not appear, repeat steps 9 and 10. If you are not successful after three tries, contact your MINC Customer Support Center.

# CHAPTER 8
# SYSTEM SPECIFICATIONS

## MiniMINC CHASSIS

### Mechanical

#### Overall dimensions

20.08″ (51 cm) long by 13″ (33 cm) wide
by 13.42″ (34.1 cm) high

#### Weight

55 lbs (25.75 kilos)

### Environmental

#### Temperature

Operating

15° to 32° C (59° to 90° F) ambient.

Temperature is derated 1.8° C per 1000 m or 1° F per 1000 ft for altitude.

Maximum gradient: 11° C per hr (20° F per hr).

Nonoperating

-35° to +60° C

(-30° to +140° F)

### Humidity

Operating

Maximum wet bulb: 25°C (77°F)

Minimum dew point: 2°C (36°F)

Relative Humidity: 20% to 80%

Nonoperating

Relative Humidity: 5% to 95% (noncondensing)

## SYSTEM RELIABILITY

### Disk Life

3 million revolutions/track
with head loaded. The head contacts
5 tracks when loaded.

### Seek Error Rate

1 in 10(6) seeks

### Soft Read Error Rate

1 in 10(9) bits read

### Hard Read Error Rate

1 in 10(12) bits read

**NOTE**

The above error rates only apply to diskettes that are properly cared for. Seek error and soft read errors are usually attributable to random effects in the head/diskette interface such as dirt or dust. These errors are also sometimes attributable to electrical noise. Errors are called "soft" if they can be recovered from in ten additional tries or less. Otherwise, they are called "hard." Seek error retries should be preceded by an INITIALIZE or RESTORE command.

## DRIVE PERFORMANCE

### Data Transfer Rate (Disk to RAM Module)

32$\mu$s/16-bit word (nominal)

**Track-to-Track-Move**

10 ms/track max

**Head Settle Time**

20 ms max

**Rotational Speed**

360 r/min ±2.5%; 166 ms/rev nominal

**Recording Surfaces per Disk**

1

**Tracks per Disk**

77 (0-76)

**Sectors per Track**

26 (1-26)

**Recording Technique**

Double frequency

**Bit Density**

3200 bits/in at inner track

**Track Density**

48 tracks/in

**Average Access**

488 ms

## POWER REQUIREMENTS

115 V/60 Hz Input: 3.0 A

90-130 V/50 Hz Input: 3.0 A

180-264 V/50 Hz Input: 2.0 A

## DEVICE SIGNALS

All device signals are serial line and compatible with EIA standard RS-232-C. Furthermore, MiniMINC requires the signal DTR (Data Terminal Ready) but does not issue DSR (Data Set Ready). MiniMINC neither requires nor provides CTS (Clear to Send) nor RTS (Request to Send).

Signal designations on 25-pin EIA connectors:

PIN 2: TERM XMIT DATA

PIN 3: TERM RCV DATA

PIN 20: TERM RDY (DTR)

PIN 1: CHASSIS GROUND

PIN 7: SIGNAL GROUND

## FLEXIBLE DISKETTES

### Description

Mylar base, oxide-coated diskette

### Dimensions

Disk: 19.8 cm (7.8 in) diameter

Jacket: 20.26 cm (7.94 in) square

### Recording Format

Single-side, industry-compatible according to IBM 3740 floppy disk format, 77 data/address tracks.

### Operating

Temperature/Humidity: Diskette temperature must be within operating range before use.

### Storage

Temperature: -34° to 52° C (-30° to 125° F)

Relative Humidity: 10% to 80% (noncondensing)

Magnetic Field: Exposure to magnetic field strength of 50 oersteds or greater may result in loss of data.

## KEYBOARD/ MONITOR WITH GRAPHICS DISPLAY (VT105)

### Alphanumeric Terminal Characteristics

* Detached keyboard with plug-in coil cord
* Screen size: 80 columns by 24 lines or 132 columns by 24 lines
* Double-width, or double-height double-width characters, selectable on a line-by-line basis
* Selectable smooth or jump scrolling
* Blinking cursor, selectable as either underline or reverse video at cursor location
* Split screen capability
* 7 x 9 dot matrix characters with descenders
* Normal or reverse video
* Nonvolatile RAM storage of terminal operating characteristics
* Single character underline, boldface, flashing, or reverse attributes accomplished on a whole screen basis whereby all selected characters are given the same attribute. The cursor position is identified by blinking the selected attribute at the cursor location.
* Composite video output — standard RS170 output for an auxiliary monitor (via BNC connector on rear of terminal)
* Built-in self-test
* P4 phosphor (white)

*Size*

* Keyboard: 45.7 cm W x 8.9 cm H x 20.3 cm D (18″ x 3.5″ x 8″)
* Combined depth: 51.4 cm (20.25″)

*Weight*

* Keyboard: 2.3 kg (5 lbs)
* Monitor: 14 kg (31 lbs)

*Video Connections*

* Composite video output (external BNC connector): Pro-

vides EIA RS170-compatible output generated by combining the video signal with a composite sync signal.

**NOTE**

Video output does not meet RS170 2-ma DC short circuit requirements.

* Output impedance: 75 ohms, DC coupled

* Sync level: 0 volts

* Black level: approx 0.3 volts with 75 ohm load

* White level: approx 1.0 volts with 75 ohm load

* Composite sync waveform: composed of 6 equalizing pulses, 6 vertical sync pulses, and 6 equalizing pulses

  Equalizing pulse width: 2.33 $\mu$sec $\pm$50 nsec

  Vertical pulse width: 27.28 $\mu$sec $\pm$200 nsec

  Horizontal pulse width: 4.71 $\mu$sec $\pm$50 nsec

  Horizontal blanking width: 11.84 $\mu$sec $\pm$50 nsec (80 column mode) 12.34 $\mu$sec $\pm$50 nsec (132 column mode)

* Front porch: 1.54 $\mu$sec $\pm$50 nsec

* Video input (external BNC connector):

  External video input signal is "OR'ed" with the internal video signal in such a way that the intensity of the beam at any point on the screen will correspond to the intensity of whichever signal would tend to make the beam brighter at that point.

  Input must sync to VT105; the VT105 will not sync to input.

  Input impedance: 75 ohms, DC coupled

  Black level: 0 volts

  White level: 1.0 volts

  Maximum continuous input: $\pm$2.0 volts

# CHAPTER 9
# OPTIONAL SERIAL ASCII EQUIPMENT

Applicable routines:

SAVE

COPY

OPEN

SET_SERIAL

MiniMINC can communicate with the following Digital Equipment Corporation printers (configured for EIA, RS-232-C, full duplex transmission):

LA35 DECwriter

LA36 DECwriter II

DECwriter IV

Attach the connector terminating the printer cable to the selected MiniMINC port as shown in Figure 7. If the connector does not mate with the PRINTER or TERMINAL ports, the printer is not configured for EIA RS-232-C transmission and will require modification by Digital Field Service.

Refer to the printer user's manual for information about printer operation.

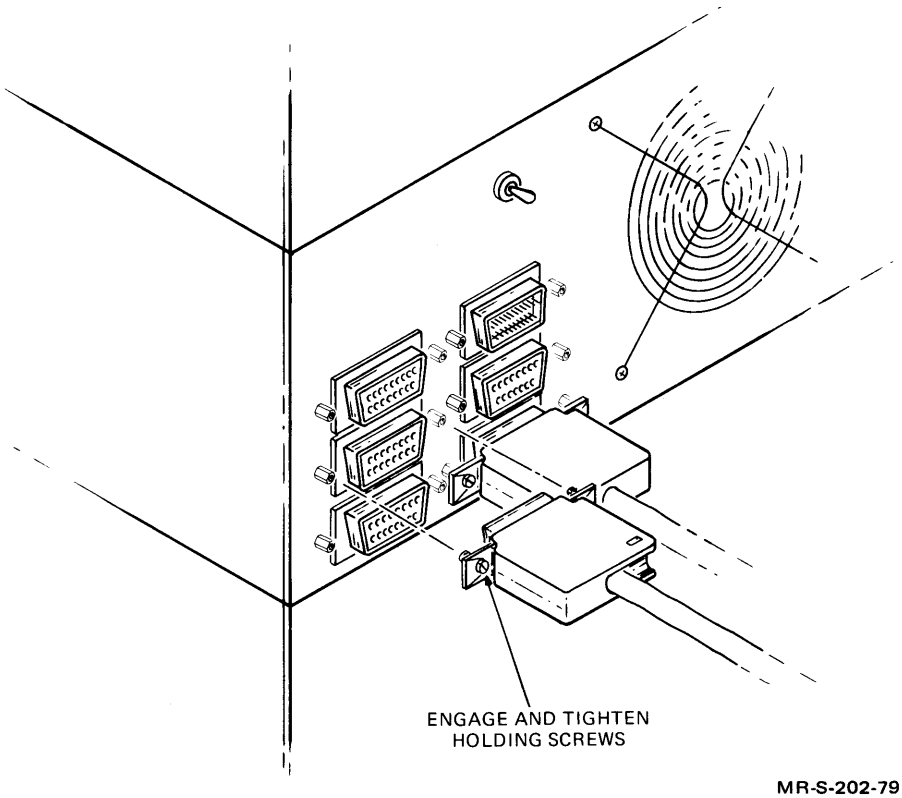



Figure 7. Attaching the Serial Input/Output Cable

# APPENDIX A
# USING THE PDT-11/150 SYSTEM EXERCISER

The PDT-11/150 system exerciser is designed to verify system performance while making extensive worst-case demands on all major components. It serves a purpose like that of the MiniMINC self-test but is more thorough and attacks more obscure problems.

The system exerciser produces output only on the console terminal. If your terminal is a VT105, you should closely monitor the output. You must be prepared to note error messages produced by the exerciser, before they scroll off the screen.

If you have access to a "hard copy" terminal, such as a DECwriter II or DECwriter IV, we recommend that you connect it temporarily to the CONSOLE port, replacing the VT105. Doing so will allow you to run the system exerciser without close observation. The resulting listing of the test results can be reviewed after completion of the test.

To run the system exerciser, proceed as follows:

1. Power down system.
2. Remove any program diskettes from drives 0 and 1.
3. Load CVKDAC PDT-11/150 System Exerciser in drive 0 (top drive).
4. Set mode switch on rear panel to NORMAL.

5. Power up system and wait for lamp 2 to cycle to its steady ON state (lamp 1 off).

6. Press SETUP key, then 5 key. Move cursor with arrow keys until located over digit 1 of group 1. Press 6 key to enter jump-scroll mode. Press SETUP key to return to normal text mode.

7. Press "@" ("SHIFT/2") on the keyboard two or three times at half-second intervals to allow the MiniMINC chassis to match the baud rate of the VT105. The system will respond with:

```
CVKDAC PDT-11/150 SYSTEM EXERCISER

SWR =  000000 NEW =
```

8. Type "112000 (RET) ". The system will respond with:

```
DEVM =  000017 NEW =
```

9. Respond to "NEW =" with "100017 (RET) ". The system will respond with the display shown in Figure 8.

```
CVKDAC  PDT-11/150 SYSTEM EXERCISER

SWR = 000000  NEW = 112000

DEVM = 000017  NEW = 100017
    30K MEMORY PRESENT
PRINTER TESTING DROPPED
PRINTER NOT PRESENT

INSERT SCRATCH DISKS, TYPE 'P' FOR NORMAL TESTING
'240G' FOR NORMAL RESTARTS
'250G' TO COPY SYS EXERCISER DISK
'260G' FOR COMPATABILITY PASS 1: WRITE
'270G' FOR COMPATABILITY PASS 2: READ
004072
@
```

MR-S-203-79

Figure 8. System Response to Configuration Input

10. Remove the system exerciser diskette and load each drive with an initialized diskette.

## CAUTION

BE SURE TO REMOVE THE SYSTEM EXERCISER DISKETTE BEFORE PROCEEDING. Be

sure also to load the drives with diskettes that do not contain useful program or data files. The system exerciser is about to write test data over the entire surface of both diskettes. It will obediently write over any diskette that happens to be in the drives when you proceed.

11. Type "SHIFT/P". The system will proceed with its testing and generate a display like that shown in Figure 9.

```
@
MEM TESTS DONE
EIS-FIS TESTS DONE
CLOCK RUNNING
SYNC COMM DONE
ASYNC COMM RUNNING
TERM #1 RUNNING
TERM #2 RUNNING
TERM #3 RUNNING
DX0 TRK      0 DONE
DX1 TRK      0 DONE
DX0 TRK      1 DONE
DX1 TRK      1 DONE
DX0 TRK      2 DONE
DX1 TRK      2 DONE
DX0 TRK      3 DONE
DX1 TRK      3 DONE
DX0 TRK      4 DONE
DX0 DATA PATT DONE
DX1 TRK      4 DONE
DX1 DATA PATT DONE
DX0 TRK     40 DONE
DX1 TRK     40 DONE
DX0 TRK     41 DONE
DX1 TRK     41 DONE
DX0 TRK     42 DONE
DX1 TRK     42 DONE
DX0 TRK     43 DONE
DX1 TRK     43 DONE
DX0 TRK     44 DONE
DX0 DATA PATT DONE
DX1 TRK     44 DONE
DX1 DATA PATT DONE
DX0 TRK     72 DONE
DX1 TRK     72 DONE
DX0 TRK     73 DONE
DX1 TRK     73 DONE
DX0 TRK     74 DONE
DX1 TRK     74 DONE
DX0 TRK     75 DONE
DX1 TRK     75 DONE
DX0 TRK     76 DONE
DX0 DATA PATT DONE
DX1 TRK     76 DONE
DX1 DATA PATT DONE
DX0 RANDOM SEEKS DONE
DX1 RANDOM SEEKS DONE

END PASS #     1        TOTAL ERRORS:          0
                        TOTAL SOFT ERRORS:     0
```

```
                                    TOTAL ERRORS THIS PASS:     0
                                    SOFT  ERRORS THIS PASS:     0
MEM TESTS DONE
EIS-FIS TESTS DONE
CLOCK RUNNING
SYNC COMM DONE
ASYNC COMM RUNNING
TERM #1 RUNNING
TERM #2 RUNNING
TERM #3 RUNNING
DX0 TRK     0 DONE
DX1 TRK     0 DONE
DX0 TRK     1 DONE
DX1 TRK     1 DONE
DX0 TRK     2 DONE
DX1 TRK     2 DONE                          MR-S-204-79
```

Figure 9. Typical System Exerciser Display

The system will execute a short pass through the disk test which, if successful, will take about 5 1/2 minutes. This will be followed by a much longer pass in which every track on both disks will be tested. If successful, this pass will take about half an hour.

If the long pass (Pass Number 2) is successful, the test is repeated indefinitely. To stop the system exerciser after a successful test, turn off the power switch on the PDT chassis and remove the scratch diskettes.

If an error is encountered during either the first or subsequent passes, the system will issue an error message, sound the bell on the console, and then halt. Write down the error message, then type "SHIFT/P" to proceed with the test.

Note that errors can be caused by two unrelated problems: faults in the disk drive(s) or bad blocks on the diskette(s). If the test reaches a point where nearly every track on one diskette generates an error halt, halt the system by depressing BREAK key and then load the diskettes into drives opposite to those in which they were previously running and repeat Step 10 by typing "240SHIFT/G". If the errors recur but are associated with the opposite disk drive, the problem is with the diskette, not the drives. Replace the defective diskette and start over. If the errors recur on the same drive as before, power down the system and contact your MINC Customer Support Center (see *MiniMINC Release Notes*).

# APPENDIX B
# VT105 INTERACTIVE GRAPHIC TEST PROCEDURE

## INTRODUCTION

The VT105 self-test procedure tests the alphanumeric features of the VT105 terminal. The VT105 self-test is discussed in the section "Troubleshooting" in this manual. The self-test does not test the graphic features of the VT105 terminal.

This appendix describes an interactive test procedure that can be used to diagnose graphic malfunctions of the VT105 terminal.

If, after running the VT105 self-test, you suspect a malfunction of the terminal's graphic hardware, follow the test procedures described in this appendix.

If one of the test procedures fails to produce the illustrated display, make a record of the test procedure that failed and of the final screen condition. Then contact your MINC Customer Support Center (see *MiniMINC Release Notes*).

Note the following rules for using the test procedures:

1. Before trying the test procedures, put your terminal in LOCAL mode and be sure that the automatic repeat feature is turned on. For details, see the section "Changing Operating Modes on the MiniMINC Terminal (Setup)" in this manual.
2. To perform a particular operation during a test procedure, you type a series of characters. The space character (generated by the space bar) is shown explicitly by the word SPACE. Unless this word appears, DO NOT type spaces; type only the characters that are shown.

3. Remember to use the SHIFT key to produce such symbols as #. The CAPS LOCK key produces upper-case letters but does not shift the nonalphabetic keys.

4. If you may a typing mistake, you can initialize the test by typing the following sequences:

   A SPACE SPACE

   I 0

   I SPACE "

After initializing the test, go back to the beginning of the test procedure and retype the required characters.

5. Within a test procedure, each operation either leaves the screen blank or produces a graphic display.

   If the result of an operation should be a blank screen, the word BLANK appears.

   If a graphic display should appear, consult the figure referred to in the operation. If the figure does not match the appearance of your screen, make sure you have typed the correct characters (that is, try the test procedure again). If repeated trials do not produce the correct result, stop, make a record of the test procedure and screen condition, and contact your MINC Customer Support Center.

6. If characters appear on the screen BEFORE you begin these test procedures, they will remain there.

## PREPARE FOR INTERACTIVE GRAPHIC TEST

Follow these procedures before beginning any other test.

| *Operation* | *Type* | *Figure* |
|---|---|---|
| Enter LOCAL mode | SETUP 4<br>SETUP | BLANK |
| [Be sure automatic repeat feature is on.] | | |
| Enter graphic mode | ESC 1 | BLANK |

| | | |
|---|---|---|
| Enter rectangular format | I SPACE SPACE | BLANK |
| Enable test | I SPACE " | BLANK |

## TEST PROCEDURES

Follow these procedures after preparing for the test.

Test Graph 0, Shaded Graph 0, and Graph 0 Brands

| *Operation* | *Type* | *Figure* |
|---|---|---|
| Enable graph 0 | A # | 10 |
| Enable shaded graph 0 | A ) | 11 |
| Enable graph 0 brands | I $ | 12 |
| Disable graph 0 brands | I SPACE | 11 |
| Disable shaded graph 0 and graph 0 | A SPACE | BLANK |

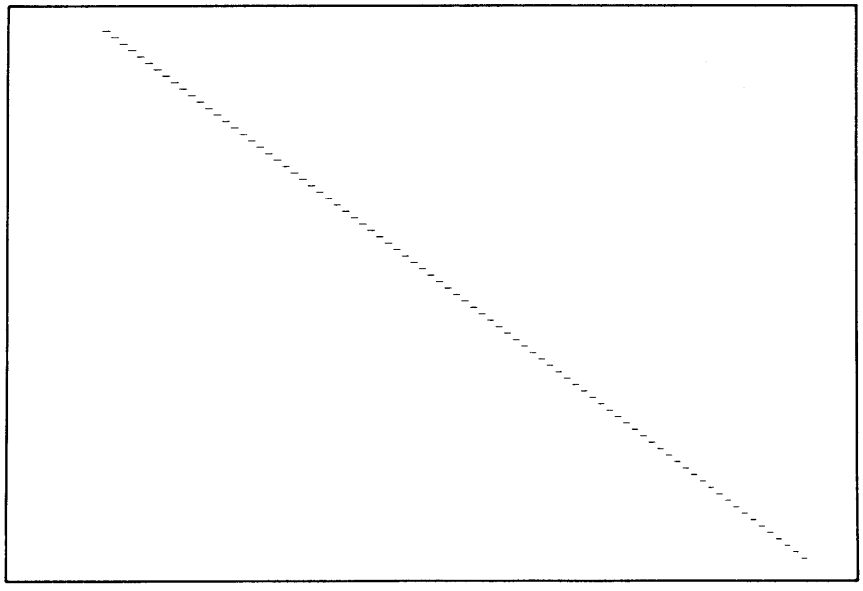
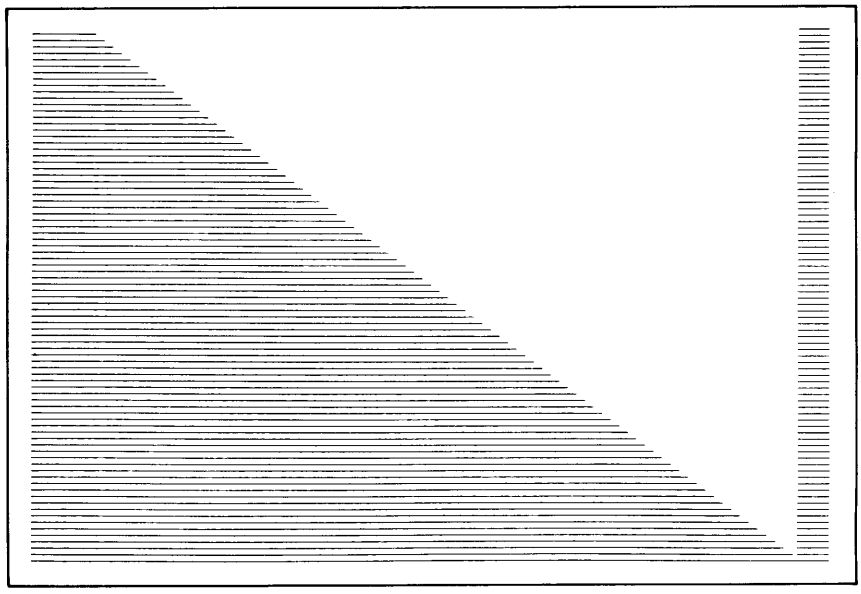


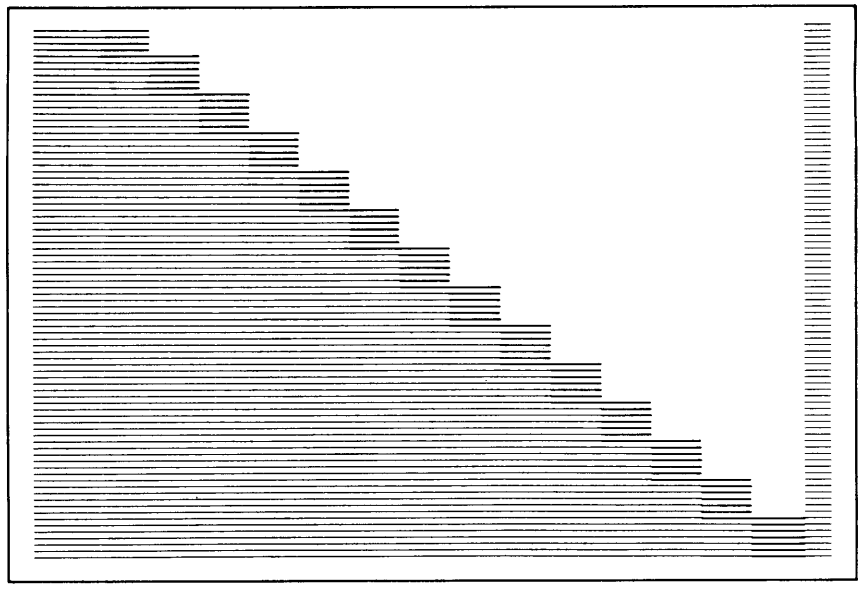
Figure 10. Graph Test Pattern

MR-S-205-79



MR-S-206-79

Figure 11. Shaded Graph Test Pattern



MR-S-207-79

Figure 12. Brand Test Pattern

Test Graph 1, Shaded Graph 1, and Graph 1 Brands

| *Operation* | *Type* | *Figure* |
|---|---|---|
| Enable graph 1 | A % | 10 |
| Enable shaded graph 1 | A 1 | 11 |
| Enable graph 1 brands | I ( | 12 |
| Disable graph 1 brands | I SPACE | 11 |
| Disable shaded graph 1 and graph 1 | A SPACE | BLANK |

Test Horizontal Lines

| *Operation* | *Type* | *Figure* |
|---|---|---|
| Enable display | A ! | BLANK |
| Enable horizontal lines | I ! | 13 |
| Disable horizontal lines | I SPACE | BLANK |

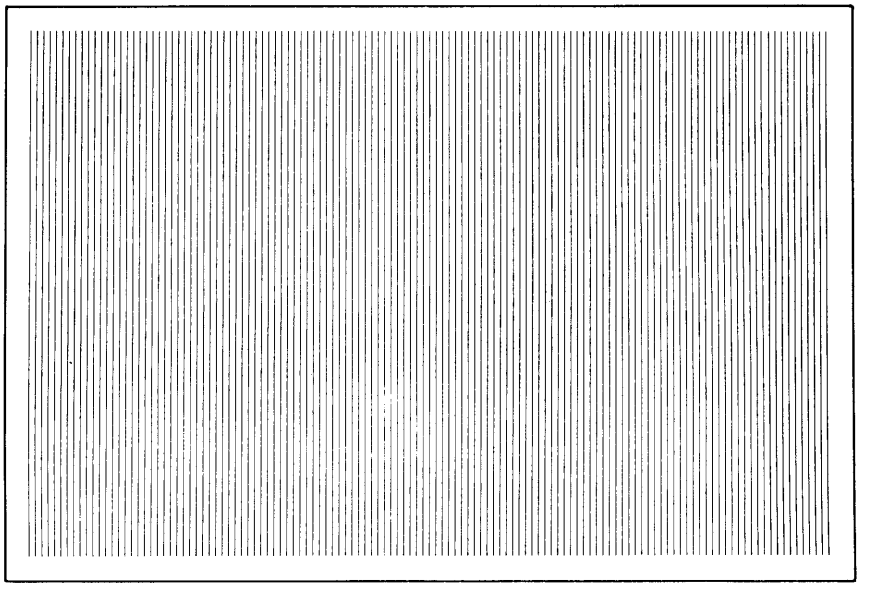


Figure 13. Horizontal Line Test Pattern MR-S-208-79

**NOTE**

The horizontal line test pattern (Figure 13) should appear to contain vertical lines.

Test Vertical Lines

| *Operation* | *Type* | *Figure* |
|---|---|---|
| Enable display | A ! | BLANK |
| Enable vertical lines | I " | 14 |
| Disable vertical lines | I SPACE | BLANK |

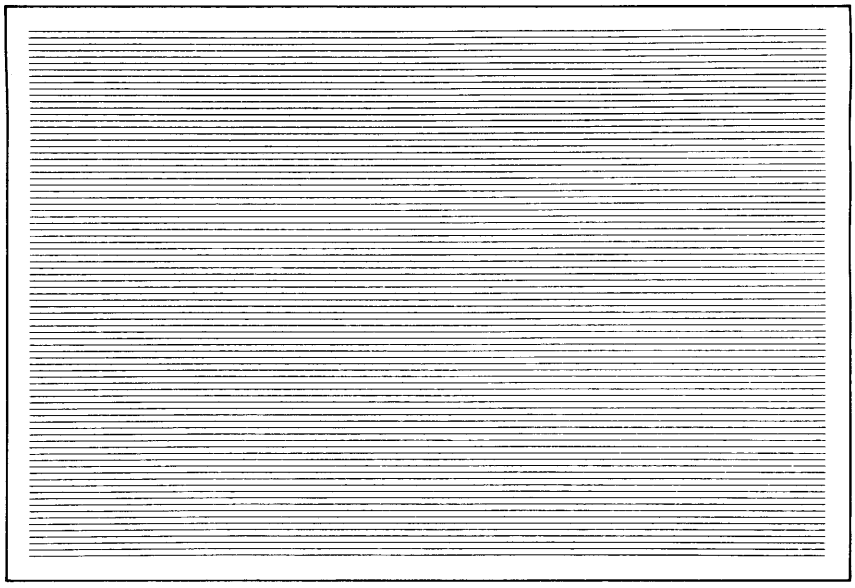


Figure 14. Vertical Line Test Pattern MR-S-209-79

**NOTE**

The vertical line test pattern (Figure 14) should appear to contain horizontal lines.

Test Shade Line 0

| *Operation* | *Type* | *Figure* |
|---|---|---|
| Enable graph 0 and shade line 0 | A # " | 11 |

| | | |
|---|---|---|
| Load shade line | @ 1 1<br>2 2<br>3 3<br>4 4<br>5 5<br>6 6<br>SPACE SPACE | <br><br>15<br><br><br><br>11 |
| Disable graph 0 and shade line 0 | A SPACE SPACE | BLANK |

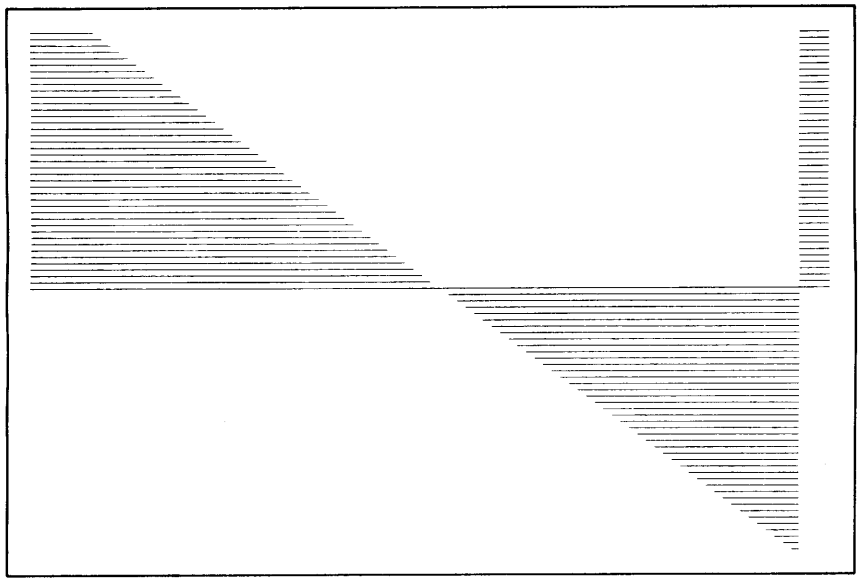


Figure 15. Shade Line Test Pattern MR-S-210-79

**NOTE**

The shade line moves upward as you perform the operation "Load shade line." Figure 15 shows the position of the shade line after you enter the characters 3 3.

Test Shade Line 1

| *Operation* | *Type* | *Figure* |
|---|---|---|
| Enable graph 1 and shade line 1 | A % % | 11 |
| Load shade line | @ 1 1<br>2 2<br>3 3 | <br><br>15 |

| | | |
|---|---|---|
| | 4 4<br>5 5<br>6 6<br>SPACE SPACE | 11 |
| Disable graph 1 and shade line 1 | A SPACE SPACE | BLANK |

Test Strip Chart 0

| *Operation* | *Type* | *Figure* |
|---|---|---|
| Enable test | I SPACE " | BLANK |
| Enable graph 0, shaded graph 0, and strip chart 0 | A + ( | 11 |
| Load X at right margin | H ? ? | 11 |
| Enable load graph 0 | B | 11 |

[Any sequence of two numbers will now cause graph to move from right to left. Holding down the SPACE bar enters zeroes, producing the display shown in Figure 16.]

| | | |
|---|---|---|
| Disable graph 0, shaded graph 0, and strip chart 0 | A SPACE SPACE | BLANK |

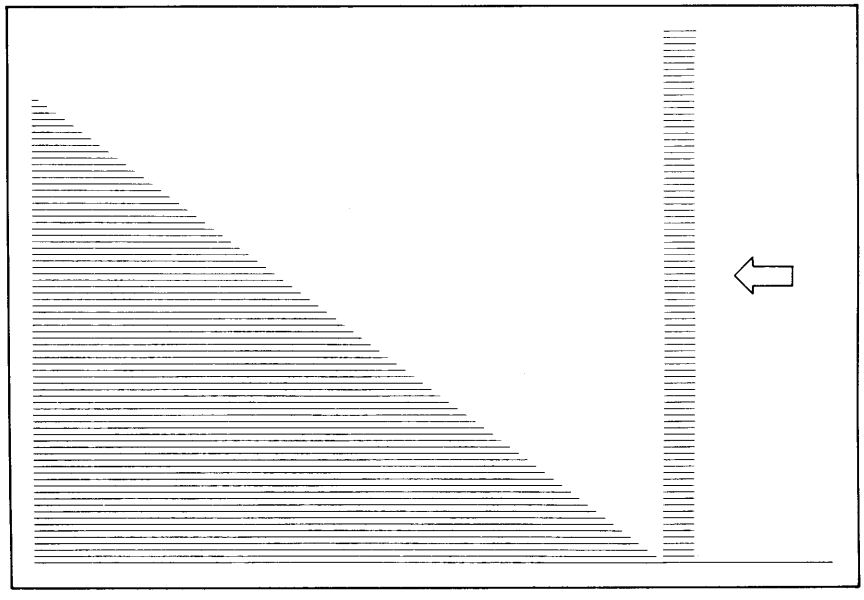


Figure 16. Strip Chart Test Pattern MR-S-211-79

**NOTE**

Figure 16 shows the result of holding down the SPACE bar after the operation "Enable load graph 0." If you hold down a different key, such as one of the numeric keys, the strip-chart action will occur with different Y values added at the right margin.

Test Strip Chart 1

| *Operation* | *Type* | *Figure* |
|---|---|---|
| Enable test | I SPACE " | BLANK |
| Enable graph 1, shaded graph 1, and strip chart 1 | A 5 ( | 11 |
| Load X at right margin | H ? ? | 11 |
| Enable load graph 1 | J | 11 |

[Any sequence of two numbers will now cause graph to move from right to left. Holding down the SPACE bar will enter zeroes, creating the display shown in Figure 16.]

| | | |
|---|---|---|
| Disable graph 1, shaded graph 1, and strip chart 1 | A SPACE SPACE | BLANK |

Exit Graphic Test

Follow these procedures at the conclusion of any graphic test, before returning to normal use of the terminal.

| *Operation* | *Type* | *Figure* |
|---|---|---|
| Exit test, initialize memories | I 0 SPACE | BLANK |
| Clear register 0 | A SPACE SPACE | BLANK |
| Clear register 1 | I SPACE SPACE | BLANK |
| Exit graphic mode | ESC 2 | BLANK |

| | | |
|---|---|---|
| Test for alpha mode | Any characters | Characters appear on screen normally |
| Exit LOCAL mode | SETUP 4 SETUP | Normal screen |

# INDEX

# 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

digital

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