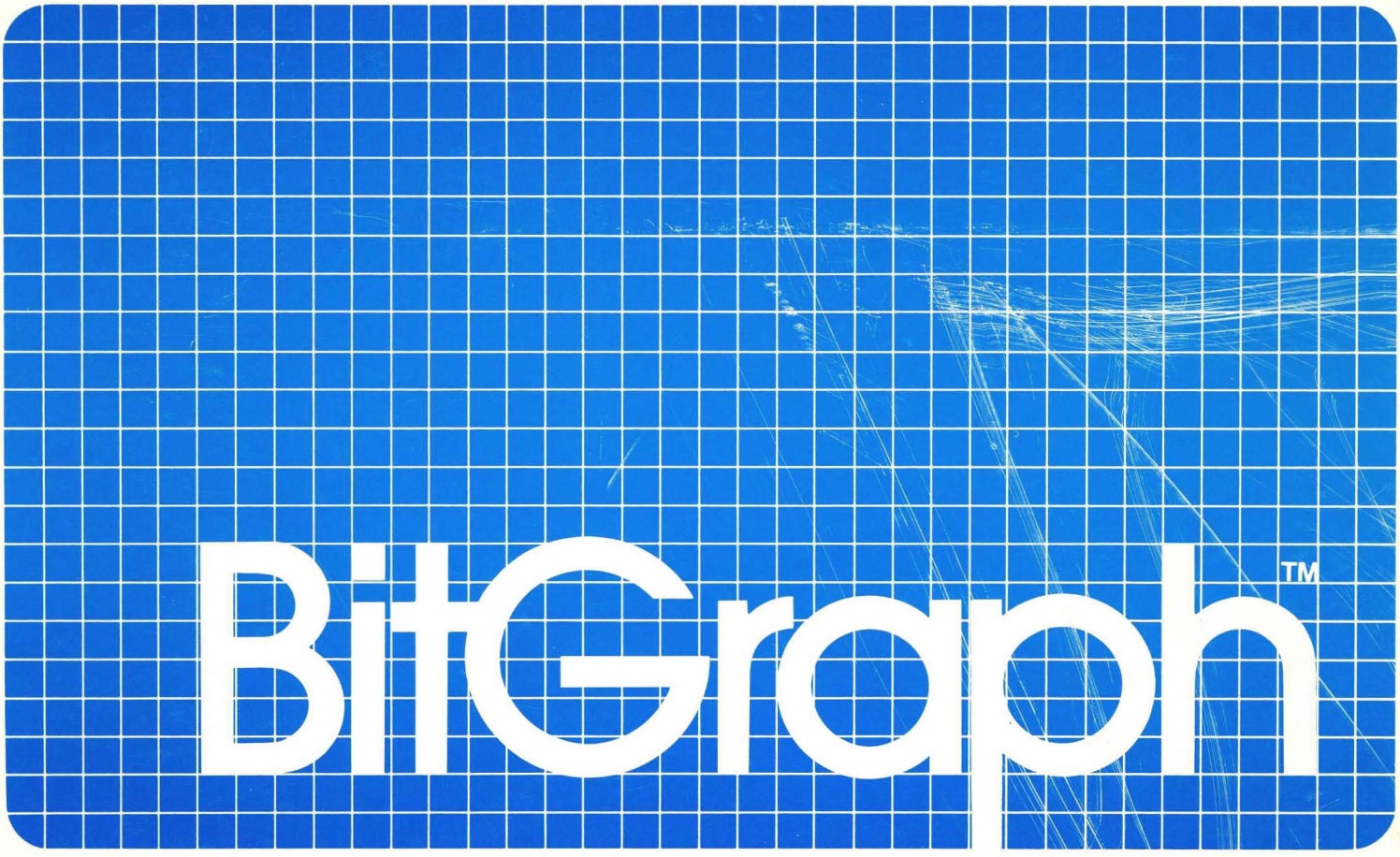
BBN Computer



Advanced Graphics Terminal

User's Guide and Operating Instructions

Document Number 0320001
Revision Number 3.0

BitGraph

Advanced Graphics Terminal

User's Guide

Version 3.2

BBN Computer Corporation

Document Number 0320001
Revision Number 3.0
May 1983

The Reader's Comments form on the last page of this publication requests your critical evaluation to help us prepare future documentation.

Printing History:
first revision, March 1982;
second revision, November 1982;
third revision, June 1983.

BBN Computer Corporation, 33 Moulton Street, Cambridge, MA 02238

CONTENTS

CONTENTS (Cont)

PREFACE

This User's Guide is a summary of the specifications and functions of Version 3.2 of the BitGraph text-and-graphics terminal. Although it contains a brief tutorial example in Chapter 3, it is primarily a reference document. Many users of the BitGraph terminal need only know how to turn the terminal on and off, and some basic setup information, and this is provided in Chapter 3. Such users will need very little of the reference information in this Guide. This is as it should be.

Chapter 1, the Introduction, gives a brief overview of the characteristics of the BitGraph terminal.

Chapter 2 presents recommended site preparation, installation, and user maintenance procedures.

Chapter 3 presents the basic features of the BitGraph terminal, and tells how to get started using it. With a simple example it indicates how BitGraph command sequences program the terminal.

The details on all the various command sequences recognized and returned by the terminal, organized by terminal emulation mode, are described in Chapter 4.

Chapter 5 presents a summary of reference material on the communications and peripheral interface options supported by the BitGraph terminal.

There are three appendixes of supplementary information and tables at the end of this Guide. Appendix A contains a summary of the ASCII character set codes. Appendix B contains a summary of the terminal command sequences, listed both alphanumerically and by functional class. The data format used to transmit packed pixel data to the terminal is defined in Appendix C.

Definitions of technical terms used throughout this manual are gathered in the Glossary.

Each of the terminal command sequences is cross-referenced in an extensive index at the end of the manual.

While all the BitGraph commands are described here, to learn how to program the BitGraph terminal or to develop MC68000 programs on a host computer and download them into it, you must use the BitGraph Programmer's Manual (BBN Computer Corporation document 032T002).

TABLES

THIS PAGE

INTENTIONALLY

LEFT BLANK

CHAPTER 1

INTRODUCTION

The BitGraph terminal combines the reliability and cost effectiveness of raster-scan CRT technology with a high-performance microprocessor architecture to produce a low-cost advanced graphics terminal. The BitGraph terminal was developed by BBN Computer Corporation for multimode graphics applications, where multiple variable-sized character fonts and detailed graphical information are combined on the display screen.

The BitGraph terminal stores graphical data as a direct bitmap in semiconductor memory, and its hardware automatically retrieves such data for display refresh. All other operations on the bitmap data are performed by the MC68000 processor. This is one reason the BitGraph terminal is more flexible and cost-effective in design than terminals that employ elaborate hardware to perform special functions that for many applications are neither useful nor necessary.

The BBN BitGraph terminal provides a high-resolution 1024-pixel by 768-pixel display on a large, vertically mounted 9 inch by 12 inch black-and-white screen. It allows users to load a large number of fonts into the terminal and to select from them at will, and it provides a variety of character-oriented cursor-addressing and editing commands.

The BitGraph terminal supports graphical display applications vector-drawing commands and raster operations to manipulate rectangular regions of the screen. Commands

are also provided to change a variety of modes and parameters within the terminal, such as the baud rate, the nominal size of characters, the use of flow control, or the rendition of characters and vectors on the screen.

Besides the advanced functionality provided by its native mode, the BitGraph Terminal also supports emulations of the DEC VT52, DEC VT100, and Tektronix 4010 terminals.

CHAPTER 2

INSTALLATION AND USER MAINTENANCE

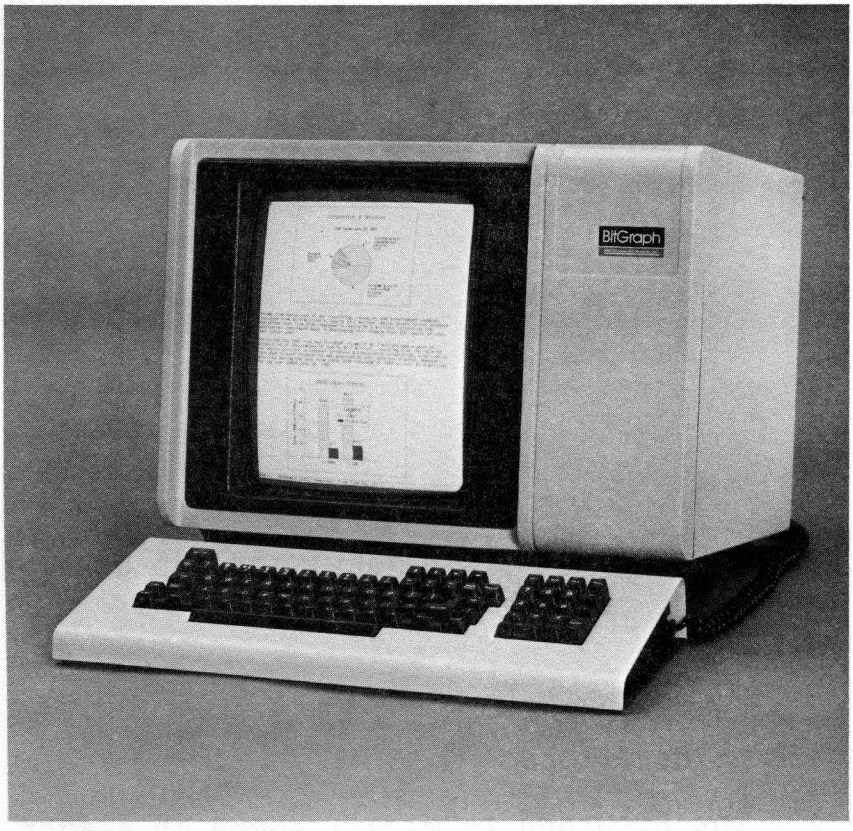
2.1 Site Considerations

In general, the BitGraph terminal, shown in Figure 2.1, can be placed anywhere that an electric typewriter might be placed. Care should be taken, however, to avoid extremes of temperature and humidity, such as from closed rooms without air-conditioning or constant exposure to direct sunlight. The terminal should not be placed in any location where it may get wet. Consult the data sheet specifications for more exact environmental considerations.

2.2 Unpacking and Installation

The BitGraph terminal shipping carton should contain the following items:

One BitGraph terminal display unit.
One BitGraph terminal keyboard unit.
One BitGraph terminal User's Guide.



To install the BitGraph terminal:

1. Remove the BitGraph terminal display unit from the shipping carton and place it in the desired work area.

2. Place the keyboard in front of the monitor and plug the keyboard interconnection cable into J4, located on the I/O connector panel at the rear of the unit. Screw the small strain relief screws on the connector into the I/O panel.

3. Verify that the on/off circuit breaker switch, located at the rear of the unit, is in the off position. Insure that a 3-prong grounded power outlet is available.

4. Connect the power cord coming from the display unit to the wall outlet.

5. Attach the host EIA cable so that the 25-pin connector mates with J6 on the I/O connector panel. Again, strain relief should be provided by the small screws on the 25-pin connector. If communications protocol other than EIA RS-232-C is to be used, consult the Interface Information for description of the necessary cables and connector assignment. NOTE: if the host computer does not assert CTS and DCD (pins 5 and 8, respectively) pin 4 must be jumpered to pin 5, and pin 8 to pin 20.

6. Depress the power switch to provide power to the terminal. After approximately one minute, you should see a banner at the top of the display, which contains a message indicating what version of the terminal firmware you have. You should also see a blinking cursor below the banner.

7. Adjust the display intensity, if necessary, by turning the remote brightness control located near the on/off switch.

8. Refer to the **Configuration Parameters** section of Chapter 4 to change any of the terminal mode settings, such as baud rate, parity, etc. The terminal is now ready to use.

2.3 User Maintenance

The keyboard keys are the only moving parts of the BitGraph terminal; they require no preventive maintenance. All of the metal and plastic BitGraph terminal surfaces may be cleaned with a soft cloth, dampened with a mild soap and water. Cleaners with solvents should not be used, as they may damage the finish. For best display quality, the surface of the display tube should be wiped occasionally with a soft, slightly damp cloth to remove fingerprints and dust. Care should be taken not to spill liquids of any sort into either the keyboard or the display unit. Such spillage can damage the circuitry.

The BBN Mouse is an optional interactive pointing device that requires little maintenance if operated correctly. The Mouse should be operated on a clean, dry surface, and care should be taken not to touch or otherwise interfere with the action of the mechanical ball bearing located on the bottom of the Mouse.

CHAPTER 3

TUTORIAL

If any terms in this Guide are unfamiliar to you, please refer to the glossary at the end, following the Appendixes.

3.1 Overview: I/O and Character Interpretation

The BitGraph terminal is an intelligent terminal used for the graphical display of visual images. The BitGraph terminal also supports a number of input or output devices, such as a typewriter-like keyboard, an interactive input device (mouse), and a hardcopy printer. It is normally connected to a computer system, called the "host," or to a telecommunications system used to access one or more host computers.

The host computer typically interacts with the BitGraph by sending and receiving 8-bit characters over the RS-232 communications link. The terminal may print characters that it receives directly on the screen, or it may interpret them according to the current "mode" of the terminal.

The BitGraph terminal interprets three types of ASCII character sequences: printing characters, control characters, and control sequences.

- Normally, printing characters flow down the communications line from the host computer, and

the BitGraph terminal firmware translates them into specific font characters. The terminal then displays these characters in linear order along a horizontal line, moving the cursor to the next character position after displaying each character.

- The BitGraph terminal uses control characters to perform low-level actions such as communications link flow control. Control characters, which are nonprinting ASCII characters, are described fully in Section 4.3.

- The terminal uses control sequences to perform higher-level actions, such as loading a font, or drawing a line.

The BitGraph terminal acts upon the characters that it receives in different ways, depending upon the current state of the terminal. These terminal states include its emulation mode, the character set that it is using as the default font, and where the cursor is with respect to the limits of the screen.

3.2 Example: Drawing a Line

The BitGraph firmware acts upon commands to create graphic images. Some commands require little or no context to be acted upon. For example, a control sequence may draw a line or vector from the current position of the cursor to the display position indicated in the sequence. To draw a line from the upper left-hand corner of the screen (the "home" position) to the lower right-hand corner, place the cursor in the upper left corner by sending the Cursor Position command (CUP) with default arguments:

```
ESC [ H
```

The line is then drawn by sending the vector-draw command (BBNDRAW), with the coordinates of the lower right-hand corner:

```
ESC : 768 ; 1024 d
```

Recall that the spaces shown in the examples are NOT actually part of the command sequence and should not be sent. Note that the above two sequences and all that follow in this tutorial may be transmitted from the host computer, or typed directly by the user on the keyboard, with the BitGraph in Local mode.

3.3 Character Sets and Fonts

Other actions, such as drawing a character, depend on the character set or font being used. The terminal may use the default font, or some other specified font. When the terminal is initially powered up, the default font is the built-in font. When the BitGraph terminal receives a printable ASCII character down the communications line, it prints that character in the default font at the current cursor position, and then moves the cursor to point to the next character location. The host computer may specify that a character be displayed in another font. In such a case, the terminal displays the character in the same position, but it may update the cursor to point to a different next-character location, depending on the width of the character just printed.

In general, the BitGraph firmware allows you to omit or **default** most parameters in a reasonable fashion. For example, if you specify that an ASCII character should be printed in a loaded font, without that font having actually been loaded, the firmware will display the character in the built-in default font.

3.4 Regions

A region is a rectangle on the display that has all the characteristics of the entire BitGraph display, but may not necessarily occupy the entire display. In other words, a region is a virtual display. Region 0, the default region, describes the characteristics of the whole display. Region 0 may not be modified by a region command.

Definition of a region does not invoke any automatic screen management functions in the BitGraph firmware - it is merely a technique to cause the firmware to restrict its attention

to a smaller portion of the display. One region is always selected, and all commands to update the display refer only to the selected region of the display. The BitGraph firmware does not support overlapping windows in the normal sense. While it is legal to define regions whose rectangles overlap, the BitGraph takes no notice of effects on information that may have been displayed in regions other than the current one.

The primary function of a region is to restrict modifications of the display to pixels within its borders. All character-, vector-, and point-drawing commands are clipped so that they only affect pixels within the current region.

CHAPTER 4

PROGRAMMING GUIDE

4.1 Keyboard

The BitGraph terminal is supplied with an 83-key detachable keyboard arranged as 65 keys in typewriter layout and an 18-key auxiliary keypad. In addition, the keyboard is equipped with an audible tone generator that supplies keyclick and bell indications, and 8 light emitting diode (LED) indicators.

The actions of the nonalphanumeric keys are described below:

LOCAL

When the terminal is in **LOCAL** mode, characters typed on the keyboard are immediately interpreted by the terminal instead of being transmitted to the host computer. Characters arriving from the host computer are discarded.

ARROW KEYS

Each of the arrow keys ↑ ↓ ← → transmits a unique sequence to the host computer, as shown in the accompanying table. If the host transmits the identical sequence, or if the terminal is in LOCAL mode, the cursor is moved one character position in the indicated direction.

Cursor Key	VT52 Mode	VT100/ANSI Mode, Cursor Key Mode Reset	VT100/ANSI Mode, Cursor Key Mode Set
Up	ESC A	ESC [A	ESC O A
Down	ESC B	ESC [B	ESC O B
Right	ESC C	ESC [C	ESC O C
Left	ESC D	ESC [D	ESC O D

Table 4.1 Cursor Key Sequences

BACKSPACE

This key transmits a backspace code. When it is received, or when the terminal is in LOCAL mode, it moves the cursor one character position to the left.

BREAK

This key transmits a break signal by holding the host communication serial line in a spacing condition for approximately 0.25 second.

NO SCROLL

When first pressed, this key stops the processing and display of data from the host computer. When pressed again, processing and display resume. Nothing is transmitted to the host as a function of pressing this key. If host flow control is enabled, an XOF character will be transmitted to the host when the input buffer begins to fill if the host continues to transmit characters.

RETURN

This key transmits a carriage return code. When it is received, or when the terminal is in LOCAL mode, it moves the cursor to the left margin.

LINE FEED

This key transmits a line feed code. When it is received, or when the terminal is in LOCAL mode, it moves the cursor one character position downward. If the cursor is on the bottom margin, a scroll up operation is performed. In addition, if **Line Feed Mode** is set, the cursor will be moved to the left margin; see the **Set Mode** and **Reset Mode** commands.

TAB

This key transmits a tab code. When it is received, or when the terminal is in LOCAL mode, it moves the cursor to the next tab stop to the right.

Numeric Keypad

The numeric keypad enables the user to enter numbers in a high-speed, calculator-like fashion. The numerals and the **minus, comma,** and **period** keys send the characters inscribed on them when they are typed. The ENTER key is equivalent to the RETURN key.

In addition, the terminal may be programmed to cause each of the keys on the keypad to send a uniquely identifiable sequence of characters. Often, it is desirable to distinguish the keys on the auxiliary keypad from the corresponding keys on the typewriter keyboard. To do this, place the auxiliary keypad in **keypad application** mode. In this mode, the keys on the keypad transmit unique sequences, given in the following table. Note that these codes are compatible with Digital Equipment Corporation's VT100 and VT52 terminals. The sequences to enter and exit **keypad application** mode are given in the **DEC VT52 Emulation** section. The sequences to alter **cursor key application** and ANSI/VT52 mode are given in the ANSI section under the **Set Mode** and **Reset Mode** commands.

PF1 - PF4

Each of these keys transmits a uniquely identifiable sequence of characters. The sequence of characters will depend on which of the several keypad modes is selected.

Keypad Key	Keypad Numeric Mode	VT52 Keypad Application Mode	VT100/ANSI Keypad Application Mode
0	0	ESC ? p	ESC O p
1	1	ESC ? q	ESC O q
2	2	ESC ? r	ESC O r
3	3	ESC ? s	ESC O s
4	4	ESC ? t	ESC O t
5	5	ESC ? u	ESC O u
6	6	ESC ? v	ESC O v
7	7	ESC ? w	ESC O w
8	8	ESC ? x	ESC O x
9	9	ESC ? y	ESC O y
-(minus)	-(minus)	ESC ? m	ESC O m
,(comma)	,(comma)	ESC ? l	ESC O l
.(period)	.(period)	ESC ? n	ESC O n
ENTER	RETURN	ESC ? M	ESC O M
PF1	ESC P	ESC P	ESC O P
PF2	ESC Q	ESC Q	ESC O Q
PF3	ESC R	ESC R	ESC O R
PF4	ESC S	ESC S	ESC O S

Table 4.2 Keypad Modes

The LED indicators are currently used to distinguish between the terminal being in On Line or Local mode. The third LED (L3) is used to indicate that the keyboard is locked, which happens when the host computer sends a flow control character (XOF) telling the terminal to suspend input to the host. Usually this is shortly followed by another flow control character (XON) telling the terminal to resume input, and therefore the L3 light blinks momentarily.

If the XON character never appears, the terminal must be reset, by entering **Setup** Mode, as described below. Although this is a rare situation, printing out a binary file may cause this effect, because an XOF character can be inadvertently sent by the host.

4.2 Configuration Parameters

A number of configuration parameters can be changed by using terminal commands. The configuration may then be saved and will subsequently be restored upon application of power. Some of the parameters that can be reconfigured are the video sense (black on white or white on black), the baud rate of the host serial port, and automatic flow control with the host. The details of the user saving and restoring of configuration parameters are described here; performing these functions from the host by use of the BitGraph Terminal Control Sequences is specified in Section 4.5 under **Restore EAROM Context** (BBNRECand **Save EAROM** Context(BBNSEC).

The **SETUP** key is used to toggle between being in and out of **Setup** Mode . When entering **Setup** Mode the top portion of the display is saved in a side buffer, and is used to display the user-settable parameters that may be set and saved. The original display contents are restored when the terminal exits **Setup** Mode. (Note: If a large number of fonts are downloaded or the user RAM is otherwise full, it is possible that the attempt to store the top of the user display will fail. This will result in a blank area at the top of the display when **Setup** Mode is exited.)

When in **Setup** Mode, the mode to be altered is highlighted. The user selects which mode to alter by using the up and down cursor keys to move to other modes in the window. When the cursor is at the top or bottom of the setup window, new modes will be scrolled onto the display as appropriate.

A mode is changed when the left or right cursor keys is pressed. The new mode is displayed. A newly selected mode is not immediately installed; it takes effect when the cursor is moved to a new mode line, or when **Setup** Mode is exited.

The following characters may be typed on the keyboard in Setup Mode with the specified meanings:

0 (zero): Total restart and reset of the terminal. Identical to power-cycle.

P or p: Exit setup mode and print display. The current contents of the display are dumped as a bitmapped image to the line printer. (Assumes the Printer Option is installed, otherwise just exits.)

S: Save the current setup information in the EAROM.

R: Recall the setup information in the EAROM.

Table 4.3 defines the default settings of the parameters and modes that are accessible in **Setup** Mode.

The following is an annotated list of the parameters and modes that may be changed in **Setup** Mode, and subsequently saved under user control. The notation (Set) and (Reset) refers to the interpretation of binary parameters, which may be manipulated by the user in **Setup** Mode, or from the host computer program by Set/Reset Mode sequences. Refer to Table 4.3 for the default settings of these parameters.

LOCAL MODE: Values are "online (set)", and "local (reset)".

LOCAL ECHO: When in the Set state, the BitGraph terminal locally echos all keystrokes. When in the Reset state, no local echoing takes place. The initial default is Reset.

SCREEN: Set to "white (set)" for a white background and "black (reset)" for a black background.

WRAP: Set to "on (set)" if characters printed at the right margin should wrap to the next line of the display. If set to "off (reset)" characters will stick at the right margin.

CURSOR CHAR and **CURSOR FONT:** Select the character and font that should be used to display the cursor.

Default Value	Parameter
0, 0	Character and font used to draw the cursor.
1, 0	Character and font used to draw pointer cursor.
<CR>	Tektronix EOT setting.
BG	Emulation Mode. (BBNSEM)
18	Font assigned to G0 character set. (SCS)
18	Font assigned to G1 character set. (SCS)
Reset	Set when G1 character set is selected. (SO, SI)
Reset	Modes that are reset initially. Line Feed Mode, Cursor Key Mode, Origin Mode, Keypad Application Mode. (SM, RM, Keypad Application Mode, Keypad Numeric Mode.) Local echo off. Initial default.
Set	Modes that are set initially. Screen Mode, Auto Wrap Mode, Flow Control Mode, Keyclick Mode, Last Point Mode. (SM, RM) Local echo on.
0	Character graphic rendition. (SGR)
8	Tab stops, initially every 8 columns. (HTS, TBC)
RASTRPL	Raster operation code for characters. (BBNSTO)
RASTRPL	Point and line drawing RastOp code. (BBNSDO)
9600	Host asynchronous port baudrate setting. (BBNSHB)
0, 0	Margin X and Y settings, respectively.
768, 1024	Margin width and height settings, respectively.

EAROM Default Parameters
Table 4.3

POINTER CHAR and POINTER FONT: A number of special characters that are provided in the terminal firmware in font 0 can be used as either the pointer or standard text cursor character. The special characters are shown in Table 4.4, below. The text cursor character always blinks, and the pointer cursor does not. The arrow characters are defined by the direction they point - NW is a diagonal arrow pointing towards the north-west corner of the screen, assuming that the top is north. The cursors are generally the opposite shading from the screen background. The shadings named in Table 4.4 assume a white background; they automatically reverse on a black background screen.

Any character of any font can be used as the cursor characters. These special characters are built in only for convenience.

Character	Symbol	Name
255	--+--	Full-screen crosshair
0	■	Black box
1		none - default
2	↖	NW arrow
3	↗	NE arrow
4	↙	SW arrow
5	↘	SE arrow
6	□	White box, black border
7	+	Small crosshair

Table 4.4 Built-In Characters for Pointer Character

TERMINAL EMULATION MODE: Set to "bg", "vt52", "vt100" or "4010" to select the appropriate emulation.

DRAWING OP and **TEXT OP:** Sets the raster operation for drawing lines and text characters, respectively. Possible values are CLR, SAD, SAND, RPL, ERASE, NOP, XOR, DRAW, NSAND, SXND, COM, SOND, NS, NSOD, NSOND, and SET. See the discussion of **Rastop** for more information on the meaning of the various operations.

G0 FONT and **G1 FONT:** Sets the G0 and G1 font selection. A value of "18" selects the standard US ASCII font (font B). A value of "17" corresponds to the United Kingdom standard font (font A), which replaces the "#" character in the US ASCII font with the British pound sign.

FLOW CONTROL: Values are "on (set)" and "off (reset)".

LAST DOT: Selecting "excluded (reset)" will cause the last point of all vectors to be omitted (useful when drawing vectors in complement mode). Selecting "included (set)" will include the last point.

ORIGIN MODE: Selecting "display (reset)" means that cursor positioning and reporting will be relative to the upper left corner of the display. If "margins (set)" is selected, they will be relative to the margin settings.

RENDITION: Selects the character rendition, currently either "normal" or "reverse" video.

TABS: A marker appears on the tab line at the bottom of the Setup window at each place where a horizontal tab is set. A small cursor appears on the tab line when the primary cursor is over the TABS item, and is moved by use of the cursor keys. By hitting the TAB key on the keyboard, the user can toggle specific tab settings along the tab line.

LINE FEED: If "<LF> (reset)" is selected, the BitGraph terminal will perform an **Index** operation when it receives a Line Feed (octal code 012) control character. If "<CR><LF> (set)" is selected, both an **Index** and a **Carriage Return** operation will be peformed.

KEY CLICK: The values "on (set)" and "off (reset)" respectively enable and disable an audible tone when keyboard keys are depressed.

BAUD: Supported values are all the baud rates provided by the BitGraph terminal hardware, from 50 to 19200 baud.

PARITY: The sense of the parity code used over the host RS232 communications interface. Values are: "none", "even", and "odd".

4010 EOT: The terminator for cursor coordinate data values sent from the terminal back to the host computer in Tektronix 4010 position reports. Normally a hardware strapping option on 4010s, default is <CR>. Supported values are "<CR>", "none", and "<CR><EOT>".

4.3 Control Characters

As mentioned in the Tutorial section, the host computer transmits three types of ASCII-coded characters to the BitGraph terminal: standard ("printing") characters, control characters, and control sequences. Control characters have octal codes 000 through 037, and 177. Those control characters that cause the BitGraph terminal to take some action are described in Table 4.5 below. Some control characters ignored on input are <u>not</u> placed in the input buffer (NUL) and so may be used as line padding characters. Others, such as DEL, are placed in the input buffer but not printed by the terminal firmware. DEL is a printing character only in font 0.

NOTE

Any control character cancels the interpretation of a control sequence, if the sequence is not yet complete. Since unrecognized control codes cause no action to be taken, the ANSI **CAN** (cancel, octal code 030) command is effectively implemented.

In Table 4.5, the abbreviation AP means "active position." The active position is usually the current location of the cursor. The terminal displays in the active position the next character that is transmitted.

Control Character	Octal Code	Action
NUL	000	Ignored on input. Not stored in buffer.
ETX	003	Responds with an ACK (006) character to the host.
BEL	007	Sounds the bell.
BS	010	Moves AP left one character position. If AP is at left margin, no action occurs.
HT	011	Also TAB. Moves AP to the next tab stop. In TEK 4010 mode, move one character space right.
LF	012	Moves AP down one line. If the ANSI LFM mode is set, also moves the horizontal position to the left margin.
VT	013	In TEK 4010 mode, moves up one char. height.
CR	015	Moves AP to left edge of display. Ends Tektronix 4010 graphics output mode or GIN mode.
US		Ends Tektronix 4010 graphics output mode or GIN mode.
SO	016	Switches to the G1 character set selected by a preceding ANSI SCS sequence.
SI	017	Switches to the G0 character set selected by a preceding ANSI SCS sequence.
DC1	021	Also called XON, used to perform flow control; see Set Mode sequence.
DC3	023	Also known as XOF, for flow control.
ESC	033	Introduces a terminal control sequence.
GS	035	Enters Tektronix 4010 graphics mode.

Interpretation of Control Characters
Table 4.5

4.4 ANSI Standard Sequences

This section consists of a definition of the standard ANSI control sequences supported by the BitGraph. As described in Chapter 2 of this manual, the BitGraph has many control commands that cause it to take action other than displaying a character on the display. Through these commands, the host can command the terminal to move the cursor, change modes, or take other special actions. Except as noted, all of the sequences described in this chapter are available for use at all times, regardless of the terminal emulation mode currently selected (see **Select Terminal Emulation**).

The sequences described in this section all conform to American National Standard X3.4-1977. Refer to the Glossary for an explanation of the terms Ps and Pn as used in the parameter specifications.

CPR Cursor Position Report ESC [Pn ; Pn R

The **Cursor Position Report** sequence is transmitted by the BitGraph terminal to the host. It reports the active position by means of the parameters. The sequence has two parameters, the first specifying the vertical position and the second specifying the horizontal position. The default condition with no parameters present is equivalent to a cursor at home position.

This control sequence is solicited by the host sending a **Device Status Report** sequence with a parameter value of 6. Note that the numbering of lines depends on the state of the **Origin** Mode; see the **Set Mode** and **Reset Mode** commands.

CUB Cursor Backward ESC [Pn D

The **Cursor Backward** sequence moves the active position to the left. The distance moved is determined by the parameter. If the parameter value is zero, one, or missing, the active position is moved one position to the left. If the parameter value is N, the active position is moved N positions to the left. If an attempt is made to move the active position beyond the left margin, the active position becomes the left margin.

CUD Cursor Down ESC [Pn B

The **Cursor Down** sequence moves the active position downward. The distance moved is determined by the parameter. If the parameter value is zero, one, or missing, the active position is moved one position downward. If the parameter value is N, the active position is moved N positions downward. If an attempt is made to move the active position beyond the bottom margin, the active position becomes the bottom margin.

CUF Cursor Forward ESC [Pn C

The **Cursor Forward** sequence moves the active position to the right. The distance moved is determined by the parameter. If the parameter value is zero, one, or missing, the active position is moved one position to the right. If the parameter value is N, the active position is moved N positions to the right. If an attempt is made to move the active position beyond the right margin, the active position becomes the right margin.

CUP Cursor Position ESC [Pn ; Pn H

The **Cursor Position** sequence moves the active position to that specified by the parameters. The first parameter specifies the vertical position, the second parameter the horizontal position. A parameter value of zero or one moves the active position to the first line or column in the display, respectively. The default condition, with no parameters present, is equivalent to a cursor to home action.

The numbering of lines depends on the state of the **Origin** Mode. When the **Origin Mode** is reset, line numbering is relative to the upper left corner of the display. When the **Origin Mode** is set, line numbering is relative to the upper left corner of the current margin settings, and the cursor cannot be positioned outside of the margins. See the **Set Mode** and **Reset Mode** sequences.

CUU Cursor Up ESC [Pn A

The **Cursor Up** sequence moves the active position upward. The distance moved is determined by the parameter. If the parameter value is zero, one, or missing, the active position is moved one position upward. If the parameter value is N, the active position is moved N positions upward. If an attempt is made to move the active position beyond the top margin, the active position becomes the top margin.

DA Device Attributes ESC [Pn c
DECID Identify Terminal ESC Z

When sent from the host with a Pn of 0 or no parameter, this sequence solicits a **DA** response. The response parameters are a function of the current emulation mode setting in the BitGraph (see **Select** EmulationMode).

Emulation Mode	Sequence Sent	Notes:
Native	ESC [? 50 ; <ver> ; <mem> ; <emu> c	1
VT100	ESC [? 1 ; 0 c	2
VT52	ESC / Z	3
TEK 4010	ESC [? 50 ; <ver> ; <mem> ; <emu> c	1,4

Notes:

1. The <ver> parameter reports the major (maj) and minor (min) software revision numbers of the BitGraph terminal software as (maj * 256 + min). For example, software revision 3.04 is transmitted as "772". The <mem> parameter is the number of bytes of free memory remaining in the terminal, divided by 1024. The <emu> is the current emulation mode setting (see **Select Emulation** Mode).

2. These parameters correspond to a VT100 terminal with no options installed (see VT100 User Guide, pg. 46).

3. These parameters correspond to a VT100 terminal emulating a VT52 terminal. (see VT100 User Guide, pp. 48, 57).

4. This sequence is included even though TEK 4010s do not answer it.

DL Delete Line ESC [Pn M

The **Delete Line** sequence deletes the specified number of lines from the display. The remaining lines below the deleted area, if any, then move up the number of lines deleted. If no parameter is specified, one line is deleted. A parameter of zero or one deletes one line. The active position is unchanged.

DCH Delete Character ESC [Pn P

The **Delete Character** sequence deletes the character at the active position and possibly other adjacent characters to the right according to the parameter. Characters to the right of the deleted characters are shifted left to the active position. The vacated character positions are cleared.

A parameter value of zero or one, or a missing parameter, deletes one character. A parameter value of N deletes N characters. The active position is unchanged.

DCS Device Control String ESC P

The **Device Control String** sequence initiates a mode in which an arbitrary string can be transmitted to the BitGraph terminal. A device control string is a string of graphic and formatting characters starting with the first character after the **Device Control String** delimiter and terminating with a **String Terminator** sequence (ESC \).

In the BitGraph terminal, the **Device Control Sequence** introduces those commands that transmit a large bulk of data:

Download or Execute
Load Font Character
Display Pixel Data

The receipt of any nonformatting control character terminates the device control string, and is then interpreted normally. The **String Terminator** sequence is interpreted as a no-operation by the BitGraph terminal. In addition, as the above commands all have some form of inherent length information, they may self-terminate when they have received the proper amount of data.

DSR Device Status Report ESC [Ps n

When transmitted to the BitGraph terminal by the host, the **Device Status Report** sequence with a parameter value of 5 solicits a **Device Status Report** response from the terminal. The status of the terminal is indicated by the parameter in the response, which is 0 for no malfunctions detected.

If the **Device Status Report** sequence is transmitted to the BitGraph terminal with a parameter value of 6, the active position is reported to the host by a **Cursor Position Report** sequence.

ED Erase in Display ESC [Ps J

The **Erase in Display** sequence clears from the active position to the end of the display inclusive if the parameter is zero or missing. A parameter of one clears from the active position to the beginning of the display, inclusive. A parameter value of two clears the entire display. Table 4.6 summarizes the clearing modes.

Parameter	Erase Display Action	Erase Line Action
0	Erases from the active position to the end of the display, inclusive.	Erases from the active position to the end of the line, inclusive.
1	Erases from the active position to the beginning of the display, inclusive.	Erases from the active position to the beginning of the line, inclusive.
2	Erases the entire display.	Erases the entire line.

Table 4.6 Summary of Screen- and Line-Clearing Modes

EL Erase in Line ESC [K

The **Erase in Line** sequence clears from the active position to the end of the line inclusive if the parameter is zero or missing. A parameter of one clears from the active position to the beginning of the line, inclusive. A parameter value of two clears the entire line. A summary of the clearing modes is given in Table 4.6.

HTS Horizontal Tabulation Set ESC H

The **Horizontal Tabulation Set** sequence sets a horizontal tab stop at the active position.

NOTE

This sequence is unavailable when VT52 emulation mode is selected.

HVP Horizontal and Vertical Position ESC [f

The **Horizontal and Vertical Position** sequence is a synonym for the **Cursor Position** sequence.

ICH Insert Character ESC [Pn @

The **Insert Character** sequence inserts erased character positions at the active position according to the parameter. The characters at and to the right of the active position are shifted the specified number of character positions to the right. Characters that are shifted beyond the right margin are discarded.

A parameter value of zero or one, or a missing parameter, inserts one erased character position. A parameter value of N inserts N erased character positions. The active position is unchanged.

IL Insert Line ESC [Pn L

The **Insert Line** sequence inserts erased lines at the active line according to the parameter. The lines at and below the active position are shifted the specified number of lines downward. Lines that are shifted beyond the bottom margin are discarded.

A parameter value of zero or one, or a missing parameter, inserts one erased line. A parameter value of N inserts N erased lines. The active position is unchanged.

IND Index ESC D

The **Index** sequence moves the active position downward one line without changing the column position. If the active position is at the bottom margin, a scroll up operation is performed.

NOTE

This sequence is unavailable when VT52 emulation mode is selected.

NEL Next Line ESC E

The **Next Line** sequence moves the active position downward one line and to the left margin. If the active position is at the bottom margin, a scroll up operation is performed.

RI Reverse Index ESC M

The **Reverse Index** sequence moves the active position upward one line without changing the column position. If the active position is at the top margin, a scroll down operation is performed.

RIS Reset to Initial State ESC c

The **Reset to Initial State** sequence resets the terminal to its power up configuration. All hardware and software state is reset.

RM Reset Mode ESC [Ps ;...; Ps l

Each parameter to the **Reset Mode** sequence causes a mode to be reset in the BitGraph terminal. A mode is considered reset until it is set by the corresponding **Set Mode (SM)** command. If the first character of the parameter string is a question mark ? (octal code 77), the parameters are interpreted as private modes. Table 4.7 describes the actions resulting from resetting a given mode.

SCS Select Character Set

The **Select Character Set** sequences are used to designate one of the possible terminal fonts as the G0 or G1 character set. The sequence has a 2-character initiator that determines whether the G0 or G1 font is being reassigned, then a single terminator character that specifies which font is desired. The **Shift In (SI)** and **Shift Out (SO)** commands (see control character section, Table 4.1) are used to select the G0 and G1 character set, respectively. Subsequent text characters are displayed in the selected font.

The BitGraph terminal initially has three built-in fonts. Font B is the United States ASCII character set. Font A is the United Kingdom standard font. Font 0 is the VT100 special graphics font. The sequences to select these fonts are given in Table 4.7.

Additional fonts may be selected by first loading the font using the **Load Font Character (BBNLFC)** sequence and the **Set Font Parameters (BBNSFP)** sequence. You may load up to 79 distinct fonts into the BitGraph terminal at one time (subject to available storage). The characters '0' through '?' specify 1 of 16 possible fonts reserved as private fonts by ANSI standards; the characters '@' through '~' specify one of the 63 possible ANSI standard fonts. Although ANY font may be loaded (even the initial built-in fonts), it is recommended that application programs use the private font numbers, because additional ANSI standard fonts may be provided in future releases.

Private Parm.	Mode Mnemonic	Reset Function	Set Function
1	Cursor Key DECCKM	Cursor keys will transmit ANSI/VT100 escape sequences. If the terminal is in VT52 emulation mode, it will continue to send VT52 sequences. See Appendix C for a complete description of keypad sequences.	Cursor keys will transmit cursor key application mode sequences. See Appendix C for a complete description of keypad sequences.
2	ANSI DECANM	Resetting this mode is equivalent to using the Select Emulation Mode command to select VT52 emumiation.	Normal (non-VT52) state for this mode.
5	Screen DECSCNM	Sets the forground color to white and the background color to black.	Sets the forground color to black and the background color to white.
6	Origin DECOM	The origin for character oriented cursor addressing sequences will be the upper left corner of the display, independent of the margin settings. The new home position becomes the active position.	The origin for character oriented cursor addressing sequences will be the upper left corner of the margin settings. The cursor cannot be positioned outside of the margins. The new home position becomes the active position.
7	Auto Wrap DECAWM	Graphic characters received when the active position is the right margin will replace any previous character there.	If the active position is the right margin, the first graphic character received will be displayed but the active position will not be advanced. Subsequent graphic characters will cause the active position to first wrap around to the beginning of the next line. A scroll up operation is performed if necessary.

Parm.	Mode Mnemonic	Reset Function	Set Function
20	Line Feed LNM	Line feed (LF) characters will move the active position down one line without affecting the horizontal position. If the active position is currently the bottom margin, a scroll up operation is performed.	Line feed (LF) characters will move the active position to the left margin of the next line. If the active position is currently the bottom margin, a scroll up operation is performed.
50	Flow Cntl BBNFLWM	Disables flow control on the host serial port. If characters are sent too quickly for the BitGraph Terminal to process them, the excess characters are discarded.	Enables flow control on the host serial port by using XOFF (DC1) and XON (DC3) characters. When the host input buffer is nearly full, an XOFF character will be transmitted to the host. When the buffer subsequently is nearly emptied, an XON character is transmitted to the host.
51	Click BBNCLKM	Disables the audible tone each time a code-transmitting key is pressed on the keyboard	Enables an audible tone each time a code-transmitting key is pressed on the keyboard.
52	Last Point BBNLPM	Vectors drawn from X1,Y1 to X2,Y2 (BBNDRAW) will include the first point but will not draw a point at X2,Y2. X2,Y2 still becomes the new active position. Used to avoid having one pixel in the wrong state at the endpoint of lines drawn in complement (RASTCOM) mode.	Vectors will be drawn from X1,Y1 to X2,Y2 inclusive.
53	Local Echo	Off in Reset state: the terminal does not echo keystrokes locally.	On in Set state: the terminal echoes keystrokes locally.

Table 4.7 Set (SM) and Reset (RM) Mode Actions

The **Load Font Character** (BBNLFC) and the **Set Font Parameters** (BBNSFP) sequences each use a number to specify the effected font. To calculate the value of the ASCII character used in the **Select Character Set** sequence that corresponds to the font number used in the BitGraph terminal private font sequences, add the value of the character '0' (32 decimal) to the font number. Similarly, to go from the **Select Character Set** terminator to a font number, subtract the value of character '0'. Thus, the font numbers 0 through 15 correspond to the characters '0' through '?' (the ANSI private fonts), and 16 through 78 correspond to the characters '@' through '~' (the ANSI standard fonts).

Font characters that have not been loaded default to the standard US ASCII font; therefore all fonts initially appear to have all characters loaded. The characters default to the **initial** US ASCII font; not the potentially overloaded one.

G0 Selection	G1 Selection	Meaning
ESC (B	ESC) B	Select US ASCII standard font.
ESC (A	ESC) A	Select UK standard font.
ESC (0	ESC) 0	Select US ASCII standard font.

Table 4.8 Select Character-Set (SCS) Sequences

SGR Select Graphic Rendition ESC [Ps ;...; Ps m

The **Select Graphic Rendition** sequence invokes the graphic rendition selected by the parameter(s). All subsequent graphic characters are rendered according to the selected parameters. An empty parameter list is equivalent to selecting a zero parameter.

Currently, the only modes implemented are 0 (for attributes off) and 7 (for reverse video). In future versions, boldface (attribute 1), italics (attribute 3), and underlining (attribute 4) may be provided. To enable the user to provide for these future enhancements, all nonzero parameters will select reverse video.

SM Set Mode ESC [Ps ;...; Ps h

Each parameter to the **Set Mode** sequence causes a mode to be set in the BitGraph terminal. A mode is considered set until it is reset by the corresponding **Reset Mode (RM)** command. If the first character of the parameter string is a question mark ? (octal code 77), the parameters are interpreted as private modes. Table 4.7 describes the actions resulting from setting a given mode.

ST String Terminator ESC \

The **String Terminator** sequence is used to terminate a **Device Control String** . If it is transmitted to the BitGraph terminal when it is not in the process of interpreting a device control string, no operation occurs.

TBC Tabulation Clear ESC [Ps g

The **Tabulation Clear** sequence is used to clear previously set horizontal tab stops. If the parameter is zero or missing, this sequence clears the tab stop at the active position. If the parameter is three, all horizontal tab stops are cleared.

4.5 BitGraph Native Mode

4.5.1 General Information

The sequences described in this section all conform to American National Standard X3.4-1977, in that they are either reserved for private use by that standard, or they begin with a private initiator. They are used to invoke those features in the BitGraph terminal that are not covered by the ANSI standard.

All of the sequences described in this chapter are available for use at all times, regardless of the terminal emulation mode currently selected (see **Select Terminal** Emulation).

4.5.2 BitGraph Specification

BBNALN Screen Alignment ESC # 8

The **Screen Alignment** sequence fills the entire display with 'E' characters in the current font. Useful for screen alignment and focusing.

BBNDAC Display All Characters ESC : Ps D

The **Display All Characters** sequence can help in debugging graphics programs by enabling you to see the characters actually sent by the host.

If the parameter is nonzero, the BitGraph terminal enters a mode in which printable characters are displayed as usual, but control characters are displayed in reverse video as the corresponding upper case letter. Control characters are not given their normal control interpretation. For instance, a reverse-video bracket [represents escape. Each carriage return is displayed as a reverse video 'M', and also causes a combination of carriage returns and line feeds for readability.

If the parameter is zero or missing, this mode is exited, and the terminal returns to normal; note that this is the only control sequence recognized in this mode.

BBNDRAW Draw Line ESC : <x> ; <y> d

The **Draw Line** sequence is used to draw a line from the active position to the specified <x> and <y> position. After drawing the line, the specified <x> and <y> position becomes the new active position.

The <x> parameter specifies the horizontal position in pixel coordinates and ranges from 0 to 767. Zero is the left edge of the display. The <y> parameter specifies the vertical position in pixel coordinates and ranges from 0 to 1023. Zero is the bottom edge of the display.

Lines may be drawn using any of the sixteen possible **Rastop** operation codes. When drawing lines, the **Rastop** operation codes are interpreted as if the source operand is a constant in the foreground color. Note that, with this interpretation, the operation codes degenerate into four distinct cases:

1. drawing lines (RASTRPL, RASTDRAW, RASTSOND, RASTSET)
2. erasing lines (RASTCLR, RASTERASE, RASTNSAND, RASTNRPL)
3. complementing lines (RASTSAND, RASTXOR, RASTCOM, RASTNSOND)
4. no operation

Use the **Set Drawing Operation** command to change the line drawing mode.

To reduce the number of characters that must be transmited to the BitGraph to draw a vector, the TEKTRONIX vector drawing sequence may be used; see **Enter Graphics Mode (GS)**.

BBNDOE Download or Execute ESC P : <op> <counthigh> ; <addrlow> ; <addrhigh> E

This sequence is used to perform a variety of functions concerned with downloading user 68000 software into the terminal.

An <op> parameter of 0 (zero) requests that the terminal accept and download Motorola Exerciser records. This is compatible with previous releases of the BitGraph Terminal software.

An <op> parameter of 1 (one) requests that the terminal accept and download packed pixel format data. The <countlow> and <counthigh> parameters specify the low and high 16 bits of a 32-bit count of the number of bytes to download. The <addrlow> and <addrhigh> parameters specify the starting address of the memory to be downloaded.

CAUTION

The **BBNDOE** sequence always downloads an even number of bytes, and should always commence on an even byte. No checksumming is performed with this downloading format; use it only over reliable serial lines.

An <op> parameter of 2 requests that the specified number of bytes of memory from the specified memory address be transmitted back to the host in packed pixel data format. When this sequence is transmitted to the terminal, it first transmits a preamble describing the length of the string to be returned, as follows:

ESC P : 2 ; <countlow> ; <counthigh> E

Here, <countlow> and <counthigh> are the low and the high 16 bits of the length of the string, repectively. This sequence is followed by the contents of the string in packed data format (Appendix E). Finally, the BitGraph transmits a **String Terminator** sequence (ESC \).

An <op> parameter of 3 stores a string in the terminal, and an <op> parameter of 4 uploads a string in the terminal. Only the <count> parameter is needed to store the string; the terminal allocates space for the string on the heap. Similarly, neither the count nor the address parameter is required for opcode 4 (upload string). It simply transmits back the string previously saved by opcode 3.

When opcode 3 is used to store a string in the terminal, the BitGraph terminal first responds with a memory management report:

ESC : 0 ; <addrlow> ; <addrhigh> M

The <addrlow> and <addrhigh> parameters give, respectively, the low and high 16 bits of the address where the string will be stored. For the purposes of storing and retrieving a string, it is sufficient to ensure that either <addrlow> or <addrhigh> is nonzero; an address of zero indicates that the memory allocation for the string failed.

After receiving the indication that the memory allocation was successful, the host transmits the contents of the string to the BitGraph two bytes at a time in the packed data format (Appendix E). Following the packed data bytes, the host should transmit a string terminator (ESC \).

The BitGraph terminal responds as for opcode 2 with the string stored by the preceding store string sequence (opcode 3) when it receives the following retrieve string sequence:

ESC P : 4 E

The intent is that opcodes 3 and 4 will be used to store a load map. The terminal can save and restore information for you if an address-independent sort of hook is used to initially find out where the information is located.

BBNDPD Display Pixel Data ESC P : Ps ; Pn ; Pn s

The **Display Pixel Data** sequence enables the user to display facsimile or bitmap images by loading encoded pixel data directly into the display memory. It initiates a mode in which subsequent packed pixel data characters are accepted and written into the display.

The packed pixel data are considered to be the source operand to any of the 16 possible **Rastop** operation codes. The operation code is specified as the first parameter.

The second parameter specifies the width of the rectangular region of the display to be filled in, and the third parameter specifies the height. Filling begins at the active position and proceeds left to right, then top to bottom. At the completion of the command, the active position is unchanged.

The format of packed pixel data is described in Appendix C. The command terminates when when any nonspacing control character is received. To conform with ANSI standards, the **String Terminator** sequence (ESC \) should be sent.

Note that **ESC P** is the ANSI standard **Device-Control String** sequence.

BBNFILL Fill Rectangle

```
ESC : <opcode> ;
<xdest>  ; <ydest>  ;
<width>  ; <height> ;
<char>   ; <font>   f
```

The **Fill Rectangle** sequence is a specialized version of **Rastop** that is used to fill a rectangular region of the display with an arbitrary stipple pattern. The <opcode> parameter specifies one of the 16 possible **Rastop** operation codes as the operation to be used when filling the rectangle. The <xdest> and <ydest> parameters specify the lower left corner of the rectangle to be filled. The <width> and <height> parameters give the size of the rectangle, which is clipped to stay within the current clipping region.

The stipple is an arbitrary sized pattern taken from the specified <char> in the specified <font>; the character must first be loaded with the stipple pattern using the **Load Font Character** command.

If the <font> parameter is omitted it will default to the last font used in a **Fill Rectangle** sequence, to save the transmission time of the extra characters. A **Fill Rectangle** sequence with a 0 <width> or <height> can be used to set the initial fill font without altering the display.

The stipple pattern is automatically aligned to an even multiple of the width and height of the specified character. The adjustment insures that the stipple pattern will be properly aligned when two rectangles drawn with the same stipple pattern are abutted.

BBNLFC Load Font Character

```
ESC P : <op> ; <char> ; <font> ;
        <rwidth>  ; <rheight> ;
        <xadjust> ; <yadjust> ;
        <xupdate> ; <yupdate> ;
        <x>       ; <y>       L
```

There are two major commands for manipulating fonts. **Load Font Character** is used to download font characters from the host computer to the BitGraph. See **Set Font Parameters** (BBNSFP) to modify font, or font character parameters.

If the <op> parameter is 0, load a new raster pattern for a font character. It initiates a mode in which subsequent packed pixel data characters are accepted and stored in the specified font character. The format of packed pixel data is described in Appendix C. The <char> parameter is the character to be changed; it must be in the range of 0 to 255 (decimal). The <font> parameter is as specified under **Select Character Set** ; it must be in the range of 0 to 78. Subsequent parameters, if unspecified, acquire their values from the font defaults. The font defaults, if not set, acquire their values from the initial US ASCII font.

Note that the BitGraph terminal can directly display only the 96 characters with ASCII values of 32 through 127 in each font. The expanded range is made available to provide a convenient area for the storage of cursors or stipple patterns that may be associated with the font.

The <rwidth> parameter specifies the width of the raster image of the character in pixels. If the <rwidth> parameter is specified but <xupdate> is NOT specified, <xupdate> defaults to <rwidth>.

The <rheight> parameter specifies the height of the raster image of the character in pixels.

The <xadjust> parameter specifies the number of pixels to the left of the cursor to draw the lower left of the raster image of the character.

The <yadjust> parameter specifies that the number of pixels below the cursor to draw the lower left of the raster image of the character.

The <xupdate> parameter specifies the number of pixels to add to the X position of the cursor after drawing the character. If the <rwidth> parameter is specified but <xupdate> is NOT specified, <xupdate> defaults to <rwidth>.

The <yupdate> parameter specifies the number of pixels to add to the Y position of the cursor after drawing the character.

Filling begins at the upper left corner of the character and proceeds left to right, then top to bottom. One bits represent the foreground color; zeros represent the background color. The command terminates when any

nonspacing control character is received. To conform with ANSI standards, the **String Terminator** sequence (ESC \) should be sent.

If the <op> parameter is 1, copy a rectangle from the display into a font character. The parameters are as for Opcode 0, with the additional <x> and <y> parameters being used to specify the location on the display of the lower left hand corner of the rectangle to copy into the font character.

The BitGraph terminal stores fonts in one of two formats. The first format is more compact, but also more restrictive than the second format. Characters of both types may be intermixed in one font to minimize storage consumption. The first format is used if the width of the font character is less than 128 (or it is defaulted) and all subsequent parameters are defaulted. The overhead for this representation is 2 bytes per character.

In the second representation, all parameters are stored as 16-bit signed quantities. The overhead for this representation is 14 bytes per character.

For each font there is additional overhead associated with the range of characters that have been loaded, where the range is the difference between the indices of the highest and lowest character loaded. The overhead is 4 bytes for each character in the range.

BBNMEM Memory Management ESC : <op> ; <low16> ; <high16> M

This control sequence allocates and loads user RAM in the terminal. In this sequence, the <low16> and <high16> parameters respectively form the least and most significant 16 bits of a 32-bit number. The firmware treats the resulting 32-bit number either as a byte count or as an address, depending on the <op> parameter.

If the <op> parameter is zero the following parameters are treated as a byte count. The sequence solicits a Memory Management report of the form:

ESC : 0 ; <addrlow> ; <addrhigh> M

This report returns the address of an allocated block of memory that is at least as large as the specified byte count. The host computer may use this block of memory for any purpose, such as for downloaded programs, data storage, and so on. If the allocation fails, the report returns a zero address.

If the <op> parameter is 1 the host treats the following parameters as the address of a previously allocated block of memory. The BitGraph terminal then frees this block of memory. The terminal makes no checks to ensure that the block of memory that it is freeing was, in fact, previously allocated by a Memory Management sequence; responsibility for maintaining the integrity of the free memory pool resides with the host computer.

BBNMOV Move Graphically ESC : <x> ; <y> m

The **Move Graphically** sequence makes the specified <x> and <y> the new active position. The <x> parameter specifies the horizontal position in pixel coordinates and ranges from 0 to 767. Zero is the left edge of the display. The <y> parameter specifies the vertical position in pixel coordinates and ranges from 0 to 1023. Zero is the bottom edge of the display.

BBNPNT Draw Point ESC : <x> ; <y> p

The **Draw Point** sequence is used to alter the pixel at the specified <x> and <y> position. After drawing the point, the specified <x> and <y> position becomes the new active position.

The <x> parameter specifies the horizontal position in pixel coordinates and ranges from 0 to 767. Zero is the left edge of the display. The <y> parameter specifies the vertical position in pixel coordinates and ranges from 0 to 1023. Zero is the bottom edge of the display.

Points may be drawn using any of the 16 possible **Rastop** operation codes. When drawing points, the **Rastop** operation codes are interpreted as if the source operand is a constant in the foreground color. Note that, with this interpretation, the operation codes degenerate into four distinct cases: opcodes for drawing points (RASTRPL,

RASTDRAW,

RASTSOND, RASTSET); those opcodes used to erase points (RASTCLR, RASTERASE, RASTNSAND, RASTNRPL); opcodes for complementing points (RASTSAND, RASTXOR, RASTCOM, RASTNSOND); and no operation. The point drawing mode is changed by using the **Set Drawing Operation** sequence.

BBNPOP Pop Context ESC :]

The **Pop Context** sequence restores the terminal context in effect at the time of a preceding **Push Context** command. If there was no preceding **Push Context** command, **Pop Context** has no effect.

BBNPRNT Print ESC P : Ps P

If the parameter is zero or missing, dumps the current contents of the display to the printer.

If the parameter is nonzero, enters a mode where following characters are simply passed through to the printer instead of being processed by the BitGraph terminal. Characters are passed until a **String Terminator** sequence (ESC \) is sent to the terminal. No portion of the **String Terminator** sequence is sent to the printer.

BBNPSG Load PSG ESC : <type> ; Pn ;... P

Support for the programmable sound generator chip has been provided in this release, but may change in future releases. This description of the **Load PSG** sequence assumes familiarity with the General Instrument's AY-3-8912 programmable sound generator chip.

The <type> parameter is a request type. Request 0 is the 'load PSG registers' request. The following arguments specify the first PSG register to load, the last register to load, and the time in milliseconds of the sound effect. The remaining arguments specify the values for the registers; omitted values are zero.

Request 1 frequency sweeps a specified PSG register. For this request, arguments following <type> are the starting and ending sweep values, the duration of each frequency 'stop' in milliseconds, and the number of a register PAIR to sweep (the PSG uses two 8-bit registers to make up each of the three 12-bit tone registers). Both up and down frequency sweeps are supported.

Upon completion of the sound effect, zeros are written into PSG registers R10-R12 (the amplitude control registers) to terminate the effect. This is almost always the correct action; you either want the terminal to be silent, or you are doing an incremental register load. In order to facilitate incremental loading, a 'load' request with a delay of 0 milliseconds suppresses the write of the amplitude registers.

BBNPSH Push Context ESC : [

The **Push Context** sequence saves the current terminal context for later restoration by the **Pop Context** sequence. Multiple levels of context saving are provided, limited only by available storage in the terminal. The terminal context is the EAROM context described under **Restore EAROM Context** plus the cursor position, the graphic origin, the margin settings, and the last font used in a **Fill Rectangle** sequence. Note that the pointer position is not saved.

BBNRAST Rastop

```
ESC : <opcode> ;
<xsrc>  ; <ysrc>   ;
<xdest> ; <ydest>  ;
<width> ; <height> r
```

The **Rastop** sequence is used to manipulate rectangular regions of the display. The <opcode> parameter defines the operation to be performed between the source and destination rectangles; see Table 4.9. Note that the mnemonics are constructed in such a way that the mnemonic itself gives the boolean equation of the operation. Aliases are defined to give more intuitively obvious names to the more common operations.

The parameters <xsrc> and <ysrc> specify the lower left corner of the source rectangle, and the parameters <xdest> and <ydest> specify the lower left corner of the destination

rectangle. The <width> and <height> parameters give the size of the rectangle. The source rectangle is clipped to remain within the display, and the destination rectangle is clipped to remain within the current clipping region.

Table 4.9 Rastop (BBNRAST) Operation Codes

BBNREC Restore EAROM Context ESC : Ps R

The **Restore EAROM Context** sequence restores the context information saved in the EAROM (Electrically Alterable Read-Only Memory). If the parameter is zero or missing, the new context is that which is stored in the EAROM. If the parameter value is one, a default context is provided.

The EAROM provides the terminal with a means for retaining mode settings for long periods of time, even when the terminal is turned off. This is the context provided upon application of power. The parameters stored in the EAROM and their default values are given in Section 4.2 under **Configuration** Parameters.

BBNRESP Set Response Initiator ESC : Pn I

This control sequence allows you to specify the first character of responses that the BitGraph terminal sends back to the host. Normally, this initial character is the <ESC> character. Some programs that interpret the <ESC> character require a different initiator.

The BBNRESP control sequence sets the initial character for all BitGraph terminal responses to the host computer. The response initiator is initially the <ESC> character (27 decimal). Note that this sequence changes the initiator only for reponses solicited by some previous sequence from the host, such as Memory Mangagement reports, Pointer Position reports, and the like. It does <u>not</u> affect any of the sequences generated by the keyboard.

Opcode	Mnemonic	Equation	Meaning
0	RASTCLR RASTNSET	0	The destination is set to the background color.
1	RASTSAD RASTNERASE	s & d	The destination is set to the bitwise AND of the source and the destination.
2	RASTSAND	s & !d	The destination is set to the bitwise AND of the source and the complement of the destination.
3	RASTS RASTRPL	s	The destination is set to the source. Equivalent to a replace operation, hence RASTRPL.
4	RASTNSAD RASTERASE	!s & d	The destination is set to the bitwise AND of the complement of the source and the destination. Equivalent to erasing those bits in the destination that are set in th source, hence RASTERASE.
5	RASTD RASTNOP	d	The destination is set to the destination. Effectively no operation is performed, hence RASTNOP.
6	RASTSXD RASTXOR	s XOR d	The destination is set to the bitwise exclusive OR of the source and the destination.
7	RASTSOD RASTDRAW	s | d	The destination is set to the bitwise OR of the source and the destination. Equivalent to settin those bits in the destination that are set in the source, hence RASTDRAW.
8	RASTNSAND	!s & !d	The destination is set to the bitwise AND of the complement of the source and the complement of the destination. May also be thought of as !(s | d).
9	RASTSXND RASTNXOR	s XOR !d	The destination is set to the bitwise exclusive OR of the source and the complement of the destination.
10	RASTND RASTCOM RASTNCOM	!d	The destination is set to the complement of the destination.
11	RASTSOND	s | !d	The destination is set to the bitwise OR of the source and the complement of the destination.
12	RASTNS RASTNRPL	!s	The destination is set to the complement of the source.
13	RASTNSOD RASTNDRAW	!s | d	The destination is set to the bitwise OR of the complement of the source and the destination.
14	RASTNSOND	!s | !d	The destination is set to the bitwise OR of the complement of the source and the complement of the destination. May also be thought of as !(s & d).
15	RASTSET RASTNCLR	1	The destination is set to the foreground color.

Table 4.9 Rastop (BBNRAST) Operation Codes

BBNSDO Set Drawing Operation ESC : Ps v

The **Set Drawing Operation** sequence selects the operation used for the drawing of lines and points. The operation may be any of the 16 possible **Rastop** operation codes. When drawing lines and points, the **Rastop** operation codes are interpreted as if the source operand is a constant in the foreground color. Note that with this interpretation, the operation codes degenerate into four distinct cases: drawing (RASTRPL, RASTDRAW, RASTSOND, RASTSET); erasing (RASTCLR, RASTERASE, RASTNSAND, RASTNRPL); complementing (RASTSAND, RASTXOR, RASTCOM, RASTNSOND); and no operation.

BBNSEC Save EAROM Context ESC : S

The **Save EAROM Context** sequence saves the portion of the current context described under the **Restore EAROM Context** sequence in the EAROM. The context at the time this sequence is executed becomes the power up state of the terminal. Note that this sequence takes several seconds to execute.

BBNSEM Select Emulation Mode ESC : Ps e

The **Select Emulation Mode** sequence selects one of several terminals to emulate. If the parameter is zero or missing, the BitGraph terminal enters native mode. Parameter values of one, two, and three select emulations of VT52, VT100, and Tektronix 4010 terminals, respectively.

BBNSFP Set Font Parameters ESC : <op> ; ... L

There are two major commands for manipulating fonts. The **Set Font Parameters** sequence, described here, is used to modify (or create) fonts. The **Load Font Character** sequence, described previously, is used to download font characters.

The specific opcodes for **Set Font Parameters** are now discussed in detail.

BBNSFP Op 1: Set Fnt ESC : 1 ; <char> ; <font> ;
<rwidth> ; <rheight> ;
<xadjust> ; <yadjust> ;
<xupdate> ; <yupdate> ;
<cwidth> ; <cheight> L

Opcode 1 is used to set the defaults for a given font. The parameters are as for **Load Font** Character. The <rwidth> and <rheight> defaults can only be set if the font is empty (i.e., it has never been loaded or it has been reset). The <cwidth> and <cheight> parameters are used to calculate character cursor coordinates. That is, the <cwidth> and <cheight> parameters determine the number of pixels that commands such as up Cursor, Line Feed, Character Cursor Positioning (CUP), etc. move the cursor.

If the <char> parameter is not defaulted, set the parameters for the specified character. The character's <rwidth> and <rheight> parameters cannot be altered. Also, the character must have been loaded as a fully specified character; see **Load Font** Character.

If the specified <char> is a copy of another (see **Copy Font** below), a new copy of the character is made before changing the the parameters so that other fonts will not 'inherit' the new parameters. Note that this invalidates the statement about storage consumption in **Copy Font** .

BBNSFP Op 2: Reset Font ESC : 2 ; <char> ; <font> L

This opcode is used to reset font entries to the initial configuration and free their allocated storage. For the font A (United Kingdom character set), font B (US ASCII standard), and font 0 (special graphics font), this means to restore the original contents of the font. For user-loaded fonts, it means to reset the font to be equivalent to the initial font B, the United States ASCII font.

If both the <char> and the <font> parameter are specified, only that character is reset. If <char> is defaulted but <font> is specified, all characters in the font are reset. If <char> is specified and <font> is defaulted, the specified character in all fonts is reset. Finally, if both <char> and <font> are defaulted, all fonts are reset.

BBNSFP Op 3: Copy Font ESC : 3 ; <schar> ; <sfont> ; <dchar> ; <dfont> L

This opcode is used to copy font characters from one font to another. The <schar> and <sfont> parameters are the source character and font, respectively. Similarly, <dchar> and <dfont> are the destination character and font. If both <schar> and <dchar> are defaulted, the entire font is copied. Defaulting either <sfont> or <dfont>, or defaulting only one of <dchar> and <schar>, causes no action.

Copied characters consume only 4 bytes of additional storage per character. Note that changing the parameters of a font character (see **Set Font Parameters** Opcode 1) causes a complete copy of the character to be made; in this case storage consumption is as described under **Load Font Character**.

BBNSFP Op 4: Select Cursor ESC : 4 ; <char> ; <font> L

This opcode selects the character used for displaying the cursor. Defaulting <char> or <font> leaves the corresponding cursor parameter unchanged. The cursor pattern is XOR'd with the display to draw the cursor. The initial cursor is character 0 in font 0.

BBNSFP Op 5: Select Pointer Cursor ESC : 5 ; <char> ; <font> L

Similar to opcode 4, but sets the character to be used as the cursor for the optional pointing device, which is typically the BBN Mouse.

The original pointing device cursor is character 1 in font 0.

BBNSHB Set Host Baudrate ESC : Ps B

The **Set Host Baudrate** sequence sets the baud rate for communication with the host to the selected rate. Ps is selected from Table 4.10. If Ps is missing, or has an illegal value, the sequence is ignored.

Ps	Baud Rate
0	50
1	75
2	110
3	134.5
4	150
5	300
6	600
7	1200
8	1800
9	2000
10	2400
11	3600
12	4800
13	7200
14	9600
15	19200

Table 4.10 Host Baud Rate Options

BBNSPP Set Pointer Parameters

```
ESC : <op> ; <flags> ;
      <btime> ; <dtime> ;
      <xpos>  ; <ypos>  ;
      <scalex>; <scaley>;
      <xincr> ; <yincr> ;
      <dx>    ; <dy>    ;
      <region>; <buttons> c
```

The **Set Pointer Parameters** sequence is used to set a variety of parameters concerning the pointing device option, generally a BBN Mouse. If the pointer option is not installed, the following operations do exactly the same thing, but the mouse or pointer never appears to move.

BNSPP Opcode 0
Set
Pointer Enable Status

```
ESC :  0     ; <flags> ;
       <btime> ; <dtime> ;
```

Set Pointer Enable
Status

```
<btime> ; <dtime> ;'''<x>     ; <y>     ;
                      <scalex>; <scaley>;
                    <region>; <buttons> c
```

This opcode is used to set the entire state of the pointing device. The bits in the <flags> parameter, if given, specify those pointer events that will be reported to the host. In addition, a bit determines if position reports are to be in absolute coordinates or in coordinates relative to the previous report.
The bits are defined as follows:

bit 0	enable right button down event.
bit 1	enable mid button down event.
bit 2	enable left button down event.
bit 3	enable right button up event.
bit 4	enable mid button up event.
bit 5	enable left button up event.
bit 6	enable button quiescent event.
bit 7	report Delta/Absolute.
bit 8	enable delta quiescent event.
bit 9	enable delta x event.
bit 10	enable delta y event.

For each enabled event a **Pointer Position Report** is sent to the host when the specified event occurs.

Each time a button changes state, the button quiescent timer is started. When <btime> milliseconds have passed with no button changing state, the button quiescent timer is halted and the button quiescent event occurs. The button event timer enables the host to detect 'chords' (i.e., holding several buttons down) and multiple button clicks.

The Report Delta / Absolute bit determines whether **Pointer Position Report** returns the (x, y) position of the pointer (when off) or the change from the previous (x, y) position reported (when on).

The <dtime> parameter sets the positioning quiescent timer. When the pointer <x> and <y> positions have not changed for <dtime> milliseconds, and the quiescent timer event is enabled, a single **Pointer Position Report** is transmitted to the host. This enables the host to determine precisely where the pointer is even if it does not cross the <dx>, <dy> threshold.

The <x> parameter sets the current X pointer position. The position is set relative to the graphics origin of the region associated with the pointer.

The <y> parameter sets the current Y pointer position. The position is set relative to the graphics origin of the region associated with the pointer.

The <scalex> and <scaley> parameters specify the distance the pointer must move (in physical pointer coordinates) before the BitGraph terminal considers it to have moved one position in x and y, respectively.

The <incrx> and <incry> parameters control the tracking of the pointer cursor. When the pointer moves <scalex> physical coordinates, <incrx> is added to the display position of the pointer. Similar processes occur for <scaley> and <incry>.

The <dx> parameter sets the threshold for sending a delta X event. Similarly, the <dy> parameter sets the threshold for sending a delta Y event.

The <region> parameter names a region (see the **Set Region Parameters** sequence) that restricts the pointer's movement. Its initial value is zero. Pointer <x> and <y> coordinates are set and reported relative to the origin of this region.

The <buttons> parameter sets the current button state.

Defaulting any parameter causes the corresponding pointer state variable to remain unchanged. When any enabled pointer event occurs the BitGraph Terminal will send an unsolicited **Pointer Position Report** .

BBNSPP Opcode 1 ESC : 1 c

This opcode solicits a **Pointer Enable Status** Report. Format is as for **Set Pointer Enable Status** ; returns the complete state of the pointing device.

BBNSPP Opcode 2 ESC : 2 c

This opcode requests the position of the pointer by soliciting a **Pointer Position Report** from the BitGraph terminal. Format of a Pointer Position Report is:

ESC : 2 ; <x> ; <y> ; <flags> c

The <x> parameter is either the change in the pointer X position or the X position relative to the origin of the region associated with the pointer. (See the **Set Pointer Enable Status** opcode.)

The <y> parameter is either the change in the pointer Y position or the Y position relative to the origin of the region associated with the pointer (see the **Set Pointer Enable Status** opcode, BBNSPP 0).

The bits in the <flags> parameter report a variety of state information:

```
bit 0:   on if right button is down.
bit 1:   on if mid   button is down.
bit 2:   on if left  button is down.
bit 3:   on if right button is up.
bit 4:   on if mid   button is up.
bit 5:   on if left  button is up.
bit 6:   on if button event timer fired.
bit 7:   on if <x> and <y> are deltas.
```

The pointer cursor is initially character 1 in font 0; see **Set Font Parameters** for details on changing this character.

BBNSRP Set Region Param. ESC : <op> ; <region> C

The **Set Region Parameters** sequence performs various operations on regions. A **region** is a rectangle that delimits the portion of the display that may be modified. Multiple regions may be defined and the **Region Switch** sequence used to switch among them. Each region has a complete copy of the terminal context described under the **Push Context** sequence, and its own context pushdown list (see **Push Context**).

All parameters to the region sequences are in absolute display coordinates. Coordinates for **Set Region Parameters** are valid in the range [-16384, +16383] (as are graphical coordinates in all BitGraph terminal sequences).

BBNSRP Op 0: Region Switch ESC : 0 ; <region> C

Switches the active cursor to the specified <region>. The <op> parameter may be defaulted to decrease transmission time. If <region> is defaulted, switches to region 0, the initial region.

BBNSRP Op 1: Region Param. Request ESC : 1 ; <region> C

Solicits a **Region Parameters Report** that returns the complete state of the specified <region>. If <region> is defaulted, returns the state of the current region.

The report has the following format:

```
ESC : 1          ; <region>  ;
                  <x>       ; <y>        ;
                  <width>   ; <height>   ;
                  <mx>      ; <my>       ;
                  <mwidth>  ; <mheight>  ;
                  <ox>      ; <oy>       C
```

The <x>, <y>, <width>, and <height> parameters are as for the **Create Region** sequence. The <mx>, <my>, <mheight>, and <mwidth> parameters give the position, width, and height of the margins for drawing characters (see the **Set Margins** opcode). The <ox> and <oy> parameters report the origin for graphic commands (see the **Set Origin** opcode).

BBNSRP Op 2: Create Region ESC : 2 ; <region> ;
<x> ; <y> ;
<width> ; <height> C

This opcode is used to create a new region.

The <region> parameter assigns a number to the new region for later reference. If the <region> parameter is defaulted a new region is allocated and assigned a unique number. A **Create Region Report** is then returned to the host computer. It passes the new region number back to the host, and has the following form:

ESC : 2 ; <region> C

The <x> and <y> parameters specify the coordinates of the lower left-hand corner of the new region. The <width> and <height> parameters specify the width and height of the new region. If any of these parameters are defaulted, they inherit the value of the corresponding region parameter in the current region.

The current **clipping region** delimits the portion of the display it is possible to modify. The boundaries of the clipping region for each region created by **Create Region** are calculated by clipping <x>, <y>, <width>, and <height> to the display.

Create Region may be applied to an existing region to modify its parameters. Defaulting a parameter leaves the corresponding region property unchanged. Note that using the **Create Region** sequence to change the position or size of a region does NOT alter the display; see **Rastop** for information on moving rectangular areas of the display. To facilitate moving the rectangular area of the display corresponding to a region, the source operand to **Rastop** is clipped to the display, instead of to the current clipping region.

Region 0 is the region that is located at (0, 0) and has a size that is equal to the size of the display. Region 0 is the display region; its position and size cannot be modified using **Create** Region.

BBNSRP Op 3: Free Region ESC : 3 ; <region> C

The specified <region> is freed. If the specified <region> is the current region, region 0 becomes the current region. Region 0 cannot be freed.

BBNSRP Op 4: Set Margins ESC : 4 ; <region> ;
<mx> ; <my> ;
<mwidth> ; <mheight> C

This sequence is used to set the margins of a given region. The margins determine the boundaries of the active position when performing character-oriented operations (e.g., drawing characters, character cursor positioning, etc.). The <mx>, <my>, <mwidth>, and <mheight> parameters specify the boundaries of the margins in a way analagous to the region boundaries in **Create** Region. A margin may be left in its current state by defaulting the corresponding margin parameter.

The margins are specified in display coordinates and are set in the specified <region>. If <region> is defaulted the margins are set in the current region. When a new region is created, the initial margin settings are inherited from the current region. The margin settings are pushed and popped by the **Push Context** and **Pop Context** sequences.

The BitGraph firmware normally computes cursor positioning with the **Cursor Position** sequence and cursor position reporting with the **Cursor Position Report** sequence relative to the upper left corner of the box determined by the margin settings. However, if **Origin Mode** is reset (set **Set Mode** and **Reset Mode** sequences), the cursor position is calculated relative to the outermost of the upper left corner of the margins and the upper left corner of the region. This admittedly complex mechanism is used to properly emulate VT100 cursor addressing and reporting functions.

Note that setting the margins outside of the boundaries of the current region may be used to establish a 'viewport' into the text being displayed, since only the portion of the text that falls within the boundaries of the current clipping region will actually be displayed. Note also that the **Set Top and Bottom Margins** sequence calculates new margins relative to the upper left corner of the current region. Finally, note that the right margin (<mx> + width>) determines the wrap position.

BBNSRP Op 5: Set Origin ESC : 5 ; <region> ; <x> ; <y> C

This opcode sets the graphics origin for the specified <region>. The <x> and <y> parameters specify the coordinate that is considered to be (0, 0) for all graphical operations. Normally, the origin is set to the same coordinates as the lower left corner of the region, but sometimes other origins are useful. For example, making the origin be the coordinates of the center of the region yields the traditional 4-quadrant Cartesian coordinate system instead of the default BitGraph Terminal coordinate system.

If <region> is defaulted the origin is set in the current region. The origin is pushed and popped by the **Push Context** and **Pop Context** sequences.

The following is a list of those operations that are relative to the graphics origin: Draw Line (BBNDRAW), Fill Rectangle (BBNFILL), Load Font Character (BBNLFC), Move Graphically (BBNMOV), Draw Point (BBNPNT), Rastop (BBNRAST), Set Pointer Parameters (BBNSPP), and Enter Graphics Mode (4010 GS).

BBNSTO Set Text Operation ESC : Ps o

The **Set Text Operation** sequence sets the operation used for displaying text characters. The parameter is one of the sixteen possible **Rastop** operation codes.

BBNSTP Set Text Parameters ESC : <width> ; <height> ; <return> ; <base> ; <wrap> b

This sequence is no longer implemented. We have replaced it with the **Set Font Parameters** and **Set Region Parameters** sequences.

4.6 DEC VT100 Emulation

The DEC VT100 terminal has several features that are not covered by ANSI standards. This section describes those features of the VT100 that are reproduced in the BitGraph terminal in order to provide a viable emulation of the VT100. All of the sequences described in this section are available for use at all times, regardless of the terminal emulation mode currently selected (see **Select Terminal Emulation**).

4.6.1 General Information

The major change to the BitGraph terminal when VT100 emulation mode is selected is to set the right margin to column 80 and the bottom margin to line 24. The features of the VT100 are supported at all times.

The BitGraph terminal does not support the following VT100 features:

Auto Repeat Mode.
Column Mode (80/132 columns).
Scrolling Mode (jump/smooth scrolling).
Return Double Height/Double Width lines to SH/SW.

4.6.2 VT100 Specification

DECRC — Restore Cursor — ESC 8

The **Restore Cursor** sequence has the same effect as the BitGraph terminal **Pop Context** sequence. Note that considerably more state information is stored than for the VT100 terminal.

DECSC — Save Cursor — ESC 7

The **Save Cursor** sequence has the same effect as the BitGraph Terminal **Push Context** sequence. Note that considerably more state information is stored than for the VT100 terminal.

DECSTBM — Set Top/Bottom Margins — ESC [<top> ; <bottom> ; <left> ; <right> r

The **Set Top and Bottom Margins** sequence sets the margins to define a scrolling rectangle. The rectangle defined by **Set Top and Bottom Margins** is the only portion of the display that will be scrolled by sequences such as **Line** Feed, Index, **Reverse** Index, **Insert** Line, and so on.

The <top> parameter is the line number of the first line in the scrolling rectangle; the <bottom> parameter is the line number of the bottom line in the scrolling rectangle. The <left> and <right> parameters represent an enhancement of the VT100 standard sequence; they specify the first and last column in the scrolling rectangle. The cursor is placed in the home position within the new margins. The definition of the home position depends on the state of the **Origin Mode** ; see the **Set Mode** and **Reset Mode** sequences.

The behavior of defaulted parameters is dependent on the current emulation mode setting (see the **Select Emulation Mode** sequence). When emulating a VT100 or VT52 terminal, defaulted <top>, <bottom>, <left>, or <right> parameters will be set to 1, 24, 1, and 80, respectively. In BitGraph native mode or Tektronix emulation mode, defaulting a parameter causes the margin to be set to the corresponding edge of the current region (see the **Set Region Parameters** sequence).

Cursor positioning with the **Cursor Position** sequence, and cursor position reporting with the **Cursor Position Report** sequence are relative to the margins if **Origin Mode** is set, and are relative to the upper left corner of the current BitGraph terminal region if **Origin Mode** is clear (see the **Set Mode** and **Reset Mode** sequences).

VT100 Enter Alternate Keypad Mode ESC =

The **Enter Alternate Keypad Mode** sequence reprograms the keypad keys to send uniquely identifiable ASCII sequences. The sequences will correspond either to VT52 or VT100 keypad application mode sequences, depending on the state of the **ANSI Mode** . See the section on the **Set Mode** (SM) command for altering the state of **ANSI Mode** . The codes transmitted are given in Section 4.1.

VT100 Exit Alternate Keypad Mode ESC >

The **Exit Alternate Keypad Mode** sequence programs the keypad keys to send the ASCII codes corresponding to the characters engraved on them.

VT100 Double Height Line (Top) (DECDHL) ESC # 3

The **Double Height Line (Bottom)** sequence marks the current line as double height (top half). In addition, this means the line is double width. Any text on the line is expanded to double height-width. The cursor properties for this line are as for double width. The double height property scrolls with the line when the display is scrolled.

VT100 Double Height Line (Bottom) (DECDHL) ESC # 4

The **Double Height Line (Bottom)** sequence marks the current line as double height (bottom half). In addition, this means the line is double width. Any text on the line is expanded to double height-width. The cursor properties for this line are as for double width. The double height property scrolls with the line when the display is scrolled.

VT100 Single Width Line (DECSWL) ESC # 5

The **Single Width Line** sequence clears the double height and double width properties of the current line. In addition, the line is cleared.

NOTE

when the VT100 returns a line to single height, completely formed single height characters are drawn on the left half of the display. The BitGraph terminal does not have the character information to be able to restore the characters; therefore, executing DECSWL on a line will clear the line.

VT100 Double Width Line (DECDWL) ESC # 6

The **Double Width Line** marks the current line as having the double width property. The current contents of the line are doubled in width (the contents of the right half of the line being lost). The double width property scrolls with the line when the display is scrolled.

VT100 Request Terminal Parameters ESC [Ps x;
DECREQTPARM

The **Request Terminal Parameters** sequence solicits a **Report Terminal Parameters** response from the BitGraph terminal. If Ps is 0 or unspecified, a type 2 report is solicited; a Ps of 1 solicits a type 3 report. A report returns as follows:

```
ESC [ <sol>    ; <par>    ; <nbits>  ;
      <xspeed> ; <rspeed> ; <clkmul> ;
                            <flags>  x
```

The meanings of these parameters are described in Table 4.11. Note that on the VT100, type 0 requests enable the terminal to send back unsolicited reports when something

changes, type 1 requests restrict the terminal to sending reports only upon request. In practice, the VT100 never sends unsolicited reports; neither does the BitGraph terminal.

Parameter	Value	Meaning
<sol>	2 or 3	A response to a request of type 0 or 1.
<par>	1	No parity specified.
	4	Parity specified and odd.
	5	Parity specified and even.
<nbits>	1	8 bit characters.
	2	7 bit characters.
<xspeed>	0	50 baud.
	8	75 baud.
	16	110 baud.
	24	134.5 baud.
	32	150 baud.
	48	300 baud.
	56	600 baud.
	64	1200 baud.
	72	1800 baud.
	80	2000 baud.
	88	2400 baud.
	96	3600 baud.
	104	4800 baud.
	108	7200 baud.*
	112	9600 baud.
	120	19200 baud.
<clkmul>	1	is always one.
<flags>	0	is always zero.

* Note that the VT100 does not operate at 7200 baud, hence the 7200 baud speed (108) is never returned by that terminal.

Table 4.11 Description of Request-Terminal Parameters

VT100 Load LEDS (DECLL) ESC [Ps q

The **Load LEDS** sequence loads the four programmable Keyboard lights according to the parameter. A zero or missing parameter clears all LEDs. Parameters 1, 2, 3, and 4 light LED 5, the left-most LED, and parameters 6, 7, and 8 light the right-most LED.

4.7 DEC VT52 Emulation

This section describes those features of the DEC VT52 that have been reproduced in the BitGraph terminal in order to provide a viable emulation of the VT52. The DEC VT52 terminal does not conform to ANSI standards; some of the sequences described in this section are available only when VT52 emulation is selected, as they collide with ANSI standard sequences.

When VT52 emulation mode is selected (see the **Select Emulation Mode** sequence), the right margin is set to column 80 and the bottom margin to line 24. The **Auto Wrap Mode** is reset so the cursor will stick in the right margin. The **Line Feed Mode** is reset. The following control sequences are rebound (acquire new meanings) when VT52 emulation is selected:

ESC D, Index.
ESC H, Horizontal Tabulation Set.

4.7.1 DEC VT52 Specification

VT52 Cursor Up ESC A

The **Cursor Up** sequence moves the active position one character position upward, without altering the horizontal position. If the active position is currently the top margin, no action occurs.

VT52 Cursor Down ESC B

The **Cursor Down** sequence moves the active position one character position downward without altering the horizontal position. If the active position is currently the bottom margin, no action occurs.

VT52 Cursor Right ESC C

The **Cursor Right** sequence moves the active position one character position to the right, without altering the vertical position. If the active position is currently the right margin, no action occurs.

VT52 Cursor Left ESC D

The **Cursor Left** sequence moves the active position one character position to the left, without altering the vertical position. If the active position is currently the left margin, no action occurs.

NOTE

This sequence is only available when VT52 emulation mode is selected.

VT52 Enter Graphics Mode ESC F

The **Enter Graphics Mode** sequence selects the graphics font. Note that the font is the VT100 special graphics font.

VT52 Exit Graphics Mode ESC G

The **Exit Graphics Mode** sequence selects the standard ASCII character set font.

VT52 Cursor to Home ESC H

The **Cursor to Home** sequence moves the active position to the top left corner of the display.

NOTE

This sequence is only available when VT52 emulation mode is selected.

VT52 Reverse Line Feed ESC I

The **Reverse Line Feed** sequence moves the active position one character position upward, without altering the horizontal position. If the active position is the top margin, the current line and all following lines are scrolled downward one position. The active position will remain unchanged; lines shifted below the bottom margin display will be lost.

VT52 Erase to End of Screen ESC J

The **Erase to End of Screen** sequence erases from the active position to the end of the display, inclusive. The active position is not changed.

VT52 Erase to End of Line ESC K

The **Erase to End of Line** sequence erases from the active position to the end of the line, inclusive. The active position is not changed.

VT52 Direct Cursor Address ESC Y <line> <col>

The **Direct Cursor Address** sequence moves the active position to the specified line and column. The line and column numbers are sent as ASCII codes whose values are the number plus 037 (8); e.g., 040 (8) refers to the first line or column, 050 (8) refers to the eighth line or column, etc. This method of cursor address encoding is often referred to as "blank biased."

VT52 Enter Alternate Keypad Mode ESC =

The **Enter Alternate Keypad Mode** sequence reprograms the keypad keys to send uniquely identifiable ASCII sequences. The sequences will correspond either to VT52 or VT100 keypad application mode sequences, depending on the state of the **ANSI Mode** . See the section on the **Set Mode** (SM) command for altering the state of **ANSI Mode** . The codes transmitted are given in Appendix C.

VT52 Exit Alternate Keypad Mode ESC >

The **Exit Alternate Keypad Mode** sequence programs the keypad keys to send the ASCII codes corresponding to the characters engraved on them. The function buttons will send the codes given in Appendix C.

VT52 VT52 Enter ANSI Mode ESC <

The **Enter ANSI Mode** sequence is provided for compatibility with the DEC VT100 terminal. The effect of the command is the same as using the BitGraph terminal **Select Emulation Mode** command to select DEC VT100 emulation.

4.8 TEKTRONIX 4010 Emulation

This section describes those features of the TEKTRONIX 4010 terminal that have been reproduced in the BitGraph terminal in order to provide emulation of the 4010. The sequences described in this section are all available at all times.

4.8.1 General Information

When Tektronix 4010 emulation mode is selected (see the **Select Emulation Mode** sequence), the portion of the display used for 4010 emulation is erased, a border is drawn around it, and the active position is moved to the home position within the Tektronix portion of the display. Display of text will wrap around at the right edge of the Tektronix window, and a line feed at the bottom of the window will wrap around to the top. Carriage return will return to the left edge of the Tektronix window rather than the left edge of the display. The character drawing operation (see the **Set Text Operation** sequence) is set to **RASTDRAW** so that characters will be painted into the display. The right margin is set to column 74, and characters in the standard font are reprogrammed to be 10 pixels wide. This is consistent with 4010 terminals given the scaling applied to the Tektronix 4010 coordinate system by the Bitgraph terminal. In Tektronix coordinates, characters are 14 pixels wide and 22 pixels wide.

The Tektronix 4010 has a notion of two text columns. A line feed on the last line of the left text column puts the cursor on the top line of the right text column, and moves the left margin to the right text column. A line feed on the last line of the right text column moves the cursor to the top line of the left text column and moves the left margin to the left text column. The two margin settings are at Tektronix coordinates:

```
Left  Margin:  x =   0.
Right Margin:  x = 518.
```

Following execution of the **Select Emulation Mode** (BBNSEM) or the Tektronix **Erase Display** sequence, the left margin is **always** selected. Also, when exiting graphics mode, the left text margin is selected.

4.8.2 Tektronix 4010 Specification

4010 Erase Display ESC FF

The **Erase Display** sequence causes exactly those actions that occur when the **Select Emulation Mode** sequence is used to select Tektronix emulation. Since all programs that use the Tektronix 4010 must first clear the screen, this sequence provides a means for automatically switching to Tektronix emulation when necessary.

4010 Enter Graphics Mode GS

The GS control code (Control-], octal code 035) causes the terminal to enter Tektronix **Graphics Output** mode. Subsequent printing characters and DEL are interpreted as graphical coordinates, and a subsequent US control code (Control-_, octal code 037) or **CAN** control code (Control-X, octal code 030) terminates graphics mode. Characters with octal codes 040 through 077 are used to represent the high-order 5 bits of a 10-bit X or Y coordinate, characters with octal codes 0100 through 0137 are used to represent the low-order five bits of an X coordinate, and characters with octal codes 0140 through 0177 are used to represent the low-order five bits of a Y coordinate. The normal order of sending the coordinate bytes is High Y, Low Y, High X, and Low X.

Receipt of a Low X coordinate causes a line to be drawn from the active position to the currently specified X and Y coordinates, and the active position is moved to the current coordinates. However, the **first** vector after entering graphics mode with the GS code is not drawn, but the active position is moved. Sending a GS code inhibits the drawing of the next vector even if the BitGraph terminal is already in graphics mode; therefore it may be thought of as a 'move cursor' command.

The High Y, Low Y, and High X bytes may be omitted if they have not changed from the previous coordinates, but if the High X byte is sent, the Low Y byte must be sent and must precede it so it can be distinguished from a High Y byte. The coordinates are scaled as follows from Tektronix coordinate space to BitGraph terminal coordinates:
seq

```
BitGraph X = ((Tektronix X + 20) * 5) /  7
BitGraph Y = ((Tektronix Y + 20) * 8) / 11
```

The Tektronix graphics output mode may be used under any terminal emulation. Scaling of coordinates is performed only when 4010 emulation is in force; therefore the graphics mode sequence may be used to reduce the number of characters that need to be transmitted to the BitGraph terminal to draw figures using the **BBNDRAW** and **BBNMOV** sequences. The disadvantage of the Tektronix format is that it provides no means for specifying negative coordinates.

4010 Enquiry ESC ENQ

The Tektronix 4010 **Enquiry** sequence causes up to seven bytes to be transmitted to the host computer. The first byte is a status byte. Bits in the status byte are:

Bit	Meaning
0	Always 1 indicating no auxiliary device.
1	0 indicates left margin is at x = 0. 1 indicates left margin is at x = 512.
2	0 indicates GRAPHICS mode. 1 indicates ALPHA mode.
3	Always 0 (linear interpolator is off).
4	Always 0 (hard copy ready).
5	Always 1.

Following the status byte are 4 bytes of cursor position information, in the same format as is used in **Graphics Output** mode. The order, however, is High X, Low X, High Y, Low Y.

The reply reports the Tektronix coordinates of the character cursor if Tektronix emulation is selected, otherwise it reports Bitgraph terminal coordinates. Thus, the **Enquiry** sequence may be used as a pixel-resolution analog to the ANSI **Device Status Report** sequence.

Following the cursor information is the termination sequence selected by the setup mode parameter Tektronix Termination (displayed as 4010 EOT). Possible settings for the termination sequence are none, carriage return only, and **CR** followed by **EOT** (octal code 004). This parameter is set on the Tektronix 4010 by hardware jumpers, and its value can be altered in **Setup** Mode. It is saved in and restored from the EAROM.

If the **Enquiry** sequence is sent while in **Grapics Input** Mode, it solicits a report of the current pointer cursor position instead of the position of the alphanumeric cursor (see the **Graphics Input** sequence).

Note that the **Enquiry** sequence and its reply may be used to implement a handshaking protocol; by sending the request periodically and waiting for the reply, the host can avoid overloading the input buffer of the terminal.

4010 Graphics Input Mode ESC SUB

The **Graphics Input Mode** sequence (GIN Mode) causes the 4010 graphics input mode to be entered. Crosshairs are displayed for the pointer cursor. The pointer may then be used to move the crosshairs. When the operator strikes any keyboard key, the terminal transmits up to 7 bytes to the host computer. The first byte is the keyboard key typed by the operator. The next four bytes are the location of the GIN cursor in the same order as in the **Enquiry** response. Following is a termination sequence selected by **Setup Mode** (see the 4010 **Enquiry** sequence).

If Tektronix emulation is in force, the coordinates are reported in the Tektronix coordinate system. In other emulations, the Bitgraph terminal coordinates of the pointer cursor are reported. Also, the pointer cursor character is only forced to be crosshairs for Tektronix emulation.

The host may terminate GIN mode by requesting the status of the graphics cursor with an **Enquiry** sequence. In this case, the terminal responds with only the four bytes of GIN cursor position, followed by the termination sequence (see the **Enquiry** sequence). Any other characters sent by the host during GIN mode are lost.

CHAPTER 5

COMMUNICATIONS INTERFACES

5.1 General

The I/O connector panel at the rear of the BitGraph cabinet provides five interfaces for connection to external devices. These connectors are numbered from top to bottom as J3(through J7. In addition, a connector is provided inside the cabinet for connection of printers and other devices using a 16-bit parallel interface. Each of these interfaces is described in detail below.

The basic BitGraph terminal configuration operates with the keyboard connected to the serial interface at J4, and the host computer connected to the EIA RS-232C interface at J6. Table 5.1 is a summary of the pin assignments for all the BitGraph peripheral and communications interface connectors.

The BitGraph has a number of options that enhance the basic terminal and allow its use in a wider range of applications. These include terminal software support for the following interfaces:

Interactive Mouse Input Device Option (J7)
Centronics-compatible Printer Option (internal)

Interfaces that will be available as software-supported options in the future include a high-speed HDLC/RS422 host communications interface (J3) and a spare RS232 interface for graphics tablet or other serial I/O. Although the connector pin assignments are described for all interfaces below, the terminal software support for the options listed above is not included with the basic terminal package, and may be purchased separately or as a field-installed upgrade.

5.2 Detailed Signal Descriptions

5.2.1 Keyboard Interface

Data between the keyboard and the main monitor electronics circuit board are communicated using asynchronous, serial TTL-level signals. Signals are interpreted as described below.

KEYBTxD (Pin 1) is serial data transmitted from the main electronics board to the keyboard. This signal is positive logic oriented and is interpreted at the keyboard as setup information.

KEYBCLK (Pin 5) provides a clock signal to the keyboard from the main electronics. This signal, though currently unused by the keyboard, is provided for future keyboards that require a synchronous protocol. The rate can be programmed up to 19.2 kilobaud.

KEYBGND (Pins 2, 4) serves as signal reference potential and as a return for the keyboard +5V DC power supply.

KEYBSUP (Pin 8) provides the input power at +5 VDC to the keyboard from VCC on the main electronics circuit board.

KEYBRxD (Pin 3) is serial data from the keyboard to the main BitGraph terminal electronics in the form of ASCII-encoded keycodes. This signal is positive logic oriented.

J3 Host HDLC Port

PIN (@DE-9 conn.)	Signal
2	HSTxD+
6	HSTxD-
7	HSRxD+
8	HSRxD-
4	HSCLK+
9	HSCLK-
1,3	GND
5	(Reserved)

J4 Keyboard Port

Pin (@DE-9)	Signal
5	KEYBCLK (unused)
1	KEYBTxD
7	KEYBSPKR
8	KEYBSUP
2,4	KEYBGND
3	KEYBRxD
9	(used as key)

J5 Port A (Spare)

Pin (@DE-9)	Signal
2	PAGND
5	PADTR
1	PATxD
6	PADCD
7	PACTS
9	PARTS
3	PARxD

J6 Host EIA Port

Pin (@DB-25)	Signal
2	HTxD
3	HRxD
4	HRTS
5	HCTS
6	HDSR
7	HGND
8	HDCD
9	HCLIN+
10	HCLIN-
11	(jumper to pin 3 for current loop)
18	HCLOUT+
20	HDTR
25	HCLOUT-

J7 Mouse Interface

Pin (@DE-9)	Signal
1	Power
2	X0
3	X1
4	Y0
5	Y1
6	Ground
7	SW0
8	SW1
9	SW2

I/O Connector Pin Assignments
Table 5.1

5.2.2 Host RS232 Interface

The signals at this connector provide asynchronous host communications compatible with EIA RS-232-C at rates up 19.2 kilobaud. In addition, signals occupying unused RS232 pins provide a 20-mA current loop interface option. The particular interface chosen is determined by the cable supplied, and the standard BitGraph host cable makes use of the RS232 interface.

RS232

HTxD (Pin 2) provides serial data from the BitGraph terminal to the peripheral device.

HGND (Pin 7) provides a common ground reference potentially for all signals at the interface.

HRxD (Pin 3) provides serial data from the peripheral to the BitGraph terminal.

HDCD (Pin 8) is a control signal from the peripheral to the BitGraph terminal. This signal must be asserted for the BitGraph terminal to transmit and receive data.

HCTS (Pin 5) is a control signal from a peripheral to the BitGraph terminal. It also must be asserted for the BitGraph terminal to transmit serial data.

HRTS (Pin 4) is a control signal asserted by the BitGraph terminal prior to transmitting data. The BitGraph terminal will assert this signal, then wait for the assertion of HCTS before transmitting.

HDTR (Pin 20) is asserted by the BitGraph terminal at all times while the unit is powered on.

Current Loop

HCLIN+	pin 9	Passive current loop signal
HCLIN-	pin 10	to the BitGraph terminal.
HCLSL	pin 11	Should be connected to HRXD pin 3 for current loop.
HCLOUT+	pin 18	Passive current loop signal
HCLOUT-	pin 25	from the BitGraph terminal.

Connect HCTS (5), HDCD (8) and HDTR (20) for either RS-232 or Current Loop operation. Alternatively, connect HRTS (4) to HCTS (5), and connect HDTR (20) to HDCD (8) and HDSR (6). Of course, you can do this only if you do not want to use these signals as they were originally intended to be used in the RS-232 protocol.

5.2.3 Mouse Interface

Connector J7 provides for connection of the BBN Mouse, or other compatible mouse, to the BitGraph terminal. With the factory or field-installed Mouse Interface option, the BitGraph provides a convenient programming interface to the Mouse as described in Chapter 4 of this manual. The signals are all TTL-level, and are used by the BBN Mouse as described below.

PWR (Pin 1) provides [unfused] +5V to the BBN Mouse circuitry.

X0 (Pin 2) and X1 (Pin 3) are the X quadrature input from the BBN Mouse, and are pulses used to encode the movement of the mouse in the X-direction.

Y0 (Pin 4) and Y1 (Pin 5) are the Y quadrature input from the BBN Mouse, and are pulses used to encode the movement of the mouse in the Y-direction.

SW0 (Pin 7), SW1 (Pin 8) and SW2 (Pin 9) are switch inputs from the three buttons on the BBN Mouse, and are grounded when the switch is depressed.

Gnd (Pin 6) serves as both power and signal ground for the

BBN Mouse.

5.2.4 Printer Interface

The Printer Interface provided at connector J8 (inside the cabinet) is designed to mate with a Printronix MVP-2 or other Centronics-compatible dot matrix or line printer. The connector on the BitGraph is located on the BitGraph Electronics Board and is a 36-pin Amphenol type 57 or equivalent plug, mating with a Amphenol 36-pin type 57 or equivalent receptacle. The cable supplied with the BitGraph Printer Interface option mates to a 36-pin Amphenol No. 57-40360 or equivalent connector located at the right rear of the printer (unless otherwise specified).

Signals on the connector are TTL-level and are assigned as follows:

DATA LINE 1 (Pin 2) Data to printer (LSB)
DATA LINE 2 (Pin 3) Data to printer
...
DATA LINE 8 (Pin 9) Data to printer (MSB)

The Data Lines drive TTL logic each inputs terminated by a 1K ohm resistor to +5 volts. The logic level is asserted by the BitGraph at least one microsecond before the leading edge of Data Strobe, and held stable for at least one microsecond after the trailing edge of the strobe pulse.

DATA STROBE (Pin 1) is a low-asserted 8.7 usec long pulse from the BitGraph clocking data into the printer.

ACK (Pin 10) is a low true pulse from the printer indicating that the printer is ready for the next transfer. This pulse must be a minimum of 2.2 microseconds long.

BUSY (Pin 11) is a high true level from the printer indicating that the printer cannot receive data because of a control code being processed. Currently this signal is ignored by the BitGraph software.

PE (Pin 12) is a high true level from the printer indicating that a check condition, such as a paper jam, exists.

Select (Pin 13) is a high true level from the printer indicating that the printer is on-line and ready for data.

PI (Pin 15) is a optional Paper Instruction to the printer. Currently this signal is not used and is driven low.

Pins 31 and 32 are spare outputs from the BitGraph that are currently not used by the software, and are driven low.

Pin 34 is a spare input to the BitGraph that is currently unused.

Pins 14, 16, 19 through 30, and 33 are grounded.

Pins 17, 18, 35 and 36 are not connected.

5.2.5 Auxiliary RS232 Interface

Connector J5 is an EIA RS-232-C compatible serial I/O port. Only a limited number of necessary EIA RS-232-C signals are available at the 9-pin connector, as described below. The port can be made plug compatible with the EIA standard by means of a short adaptor cable similar to that used on the host EIA port (J7). Currently this port is unused by BitGraph terminal software.

PATxD (Pin 1) provides serial data from the BitGraph terminal to a peripheral device.

PAGND (Pin 2) provides a common ground reference potential for all signals at the interface.

PADCD (Pin 6) is interpreted by the BitGraph terminal as a signal from the DCE or other peripheral device that the channel carrier is established. It is necessary to assert this signal for the BitGraph terminal to transmit and receive data over this interface.

PACTS (Pin 7) is similar to PADCD in that it signals the BitGraph terminal that the channel can accept data. This signal must be asserted for data to be transmitted by the BitGraph.

PARTS (Pin 9) is asserted by the BitGraph terminal prior to transmitting data. The BitGraph sends this signal to the peripheral device and then examines **PACTS** before transmitting data.

PADTR (Pin 5) is asserted at all times while the BitGraph terminal is powered up.

PARxD (Pin 3) provides serial data from the peripheral to the BitGraph terminal.

5.2.6 Host HDLC/RS422

J3 provides for future synchronous-serial communications protocols to be implemented between the BitGraph terminal and a host computer. The signals conform to EIA standard RS-422. Currently, all of these signals are ignored at all times by the BitGraph software.

HSTxD+ (Pin 2) will be positive with respect to **HSTxD-** when the BitGraph terminal is transmitting a SPACE or ON (binary 0) and negative with respect to **HSTxD-** when the BitGraph is transmitting a MARK or OFF (binary 1).

HSTxD- (Pin 6) will be negative with respect to **HSTxD+** when the BitGraph terminal is transmitting a SPACE or ON (binary 0) and positive with respect to **HSTxD+** when the BitGraph is transmitting a MARK or OFF (binary 1).

HSRxD+ (Pin 7) and **HSRxD-** (Pin 8) will be interpreted by the BitGraph terminal as receiving a SPACE or ON (binary 0) when **HSRxD+** is positive with respect to HSRxD-. The BitGraph will interpret that it is receiving a MARK or OFF (binary 1) when **HSRxD+** is negative with respect to HSRxD-.

HSCLK+ (Pin 4) and **HSCLK-** (Pin 9) will be used to clock both transmitted and received data across this interface. A clock transition will be received by the BitGraph terminal for every change in polarity of **HSCLK+** with respect to HSCLK-. The supplied clock frequency can be from 0 to 200 KHz.

HSGND (Pin 1, 3). This conductor is the same potential as both AC and DC electrical ground.

APPENDIX A

ASCII CHARACTER SET

Oct	Hex	Dec	Char	Oct	Hex	Dec	Char	Oct	Hex	Dec	Char	Oct	Hex	Dec	Char
000	00	0	NUL	040	20	32	SP	100	40	64	@	140	60	96	`
001	01	1	SOH	041	21	33	!	101	41	65	A	141	61	97	a
002	02	2	STX	042	22	34	"	102	42	66	B	142	62	98	b
003	03	3	ETX	043	23	35	#	103	43	67	C	143	63	99	c
004	04	4	EOT	044	24	36	$	104	44	68	D	144	64	100	d
005	05	5	ENQ	045	25	37	%	105	45	69	E	145	65	101	e
006	06	6	ACK	046	26	38	&	106	46	70	F	146	66	102	f
007	07	7	BEL	047	27	39	'	107	47	71	G	147	67	103	g
010	08	8	BS	050	28	40	(	110	48	72	H	150	68	104	h
011	09	9	HT	051	29	41	)	111	49	73	I	151	69	105	i
012	0A	10	NL	052	2A	42	*	112	4A	74	J	152	6A	106	j
013	0B	11	VT	053	2B	43	+	113	4B	75	K	153	6B	107	k
014	0C	12	NP	054	2C	44	,	114	4C	76	L	154	6C	108	l
015	0D	13	CR	055	2D	45	-	115	4D	77	M	155	6D	109	m
016	0E	14	SO	056	2E	46	.	116	4E	78	N	156	6E	110	n
017	0F	15	SI	057	2F	47	/	117	4F	79	O	157	6F	111	o
020	10	16	DLE	060	30	48	0	120	50	80	P	160	70	112	p
021	11	17	DC1	061	31	49	1	121	51	81	Q	161	71	113	q
022	12	18	DC2	062	32	50	2	122	52	82	R	162	72	114	r
023	13	19	DC3	063	33	51	3	123	53	83	S	163	73	115	s
024	14	20	DC4	064	34	52	4	124	54	84	T	164	74	116	t
025	15	21	NAK	065	35	53	5	125	55	85	U	165	75	117	u
026	16	22	SYN	066	36	54	6	126	56	86	V	166	76	118	v
027	17	23	ETB	067	37	55	7	127	57	87	W	167	77	119	w
030	18	24	CAN	070	38	56	8	130	58	88	X	170	78	120	x
031	19	25	EM	071	39	57	9	131	59	89	Y	171	79	121	y
032	1A	26	SUB	072	3A	58	:	132	5A	90	Z	172	7A	122	z
033	1B	27	ESC	073	3B	59	;	133	5B	91	[	173	7B	123	{
034	1C	28	FS	074	3C	60	<	134	5C	92	\	174	7C	124	|
035	1D	29	GS	075	3D	61	=	135	5D	93	]	175	7D	125	}
036	1E	30	RS	076	3E	62	>	136	5E	94	^	176	7E	126	~
037	1F	31	US	077	3F	63	?	137	5F	95	_	177	7F	127	DEL

THIS PAGE

INTENTIONALLY

LEFT BLANK

APPENDIX B

COMMAND SUMMARY

Arranged by Emulation Mode

ANSI Sequences

CPR	Cursor Position Report	ESC [Pn ; Pn R
CUB	Cursor Backward	ESC [Pn D
CUD	Cursor Down	ESC [B
CUF	Cursor Forward	ESC [C
CUP	Cursor Position	ESC [Pn ; Pn H
CUU	Cursor Up	ESC [A
DL	Delete Line	ESC [Pn M
DCH	Delete Character	ESC [Pn P
DCS	Device Control String	ESC P
DSR	Device Status Report	ESC [Ps n
ED	Erase in Display	ESC [Ps J
EL	Erase in Line	ESC [K
HTS	Horizontal Tabulation Set	ESC H
HVP	Horizontal and Vertical Position	ESC [f
ICH	Insert Character	ESC [Pn @
IL	Insert Line	ESC [Pn L
IND	Index	ESC D
NEL	Next Line	ESC E
RI	Reverse Index	ESC M
RIS	Reset to Initial State	ESC c
RM	Reset Mode	ESC [Ps ;...; Ps l
SCS	Select Character Set	See Table 4.4.

SGR	Select Graphic Rendition	ESC [Ps ;...; Ps m
SM	Set Mode	ESC [Ps ;...; Ps h
ST	String Terminator	ESC
TBC	Tabulation Clear	ESC [Ps g

BitGraph Terminal Private Sequences

BBNALN	Screen Alignment	ESC # 8
BBNDAC	Display All Characters	ESC : Ps D
BBNDRAW	Draw Line	ESC : <x> ; <y> d
BBNDOE	Download or Execute	ESC P : E
BBNDPD	Display Pixel Data	ESC P : Ps ; Pn ; Pn s
BBNFILL	Fill Rectangle	ESC : <opcode> ; <xdest> ; <ydest> ; <width> ; <height> ; <char> ; <font> f
BBNLFC	Load Font Character	ESC P : <op> ; <char> ; <font> ; <rwidth> ; <rheight> ; <xadjust> ; <yadjust> ; <xupdate> ; <yupdate> ; <x> ; <y> L
BBNMEM	Memory Management	ESC : <op>; <low 16>;<high 16> M
BBNMOV	Move Graphically	ESC : <x> ; <y> m
BBNPNT	Draw Point	ESC : <x> ; <y> p
BBNPOP	Pop Context	ESC :]
BBNPRNT	Print	ESC P : Ps P
BBNPSG	Load PSG	ESC : <type> ; Pn ;... P
BBNPSH	Push Context	ESC : [
BBNRAST	Rastop	ESC : <opcode> ; <xsrc> ; <ysrc> ; <xdest> ; <ydest> ; <width> ; <height> r
BBNREC	Restore EAROM Context	ESC : Ps R
BBNRESP	Set Response Initiator	ESC : Pn I
BBNSDO	Set Drawing Operation	ESC : Ps v
BBNSEC	Save EAROM Context	ESC : S
BBNSEM	Select Emulation Mode	ESC : Ps e
BBNSFP	Set Font Parameters	ESC : <op> ; ... L
BBNSHB	Set Host Baudrate	ESC : Ps B
BBNSPP	Set Pointer Parameters	ESC : <op> ; ... c
BBNSRP	Set Region Param.	ESC : <op> ; <region> C

BBNSTO	Set Text Operation	ESC : Ps o

DEC VT100 Emulation Sequences

VT100 DECRC	Restore Cursor	ESC 8
VT100 DECSC	Save Cursor	ESC 7
VT100 DECSTBM	Set Top and Bottom Margins	ESC [Pn ; Pn r
VT100	Enter Alternate Keypad Mode	ESC =
VT100	Exit Alternate Keypad Mode	ESC >

DEC VT52 Emulation Sequences

VT52	Cursor Up	ESC A
VT52	Cursor Down	ESC B
VT52	Cursor Right	ESC C
VT52	Cursor Left	ESC D
VT52	Enter Graphics Mode	ESC F
VT52	Exit Graphics Mode	ESC G
VT52	Cursor to Home	ESC H
VT52	Reverse Line Feed	ESC I
VT52	Erase to End of Screen	ESC J
VT52	Erase to End of Line	ESC K
VT52	Direct Cursor Address	ESC Y <line> <col>
VT52	Enter Alternate Keypad Mode	ESC =
VT52	Exit Alternate Keypad Mode	ESC >
VT52	VT52 Enter ANSI Mode	ESC <

TEKTRONIX 4010 Emulation Sequences

4010	Request Terminal Status	ESC ENQ
4010	Erase Display	ESC FF
4010	Enter Graphics Mode	GS

Control Sequence Summary
Sorted Alphabetically by Escape Code

ESC \	ST	String Terminator
ESC # 3	DECDHLT	Double Height Line Top
ESC # 4	DECDHLB	Double Height Line Bottom
ESC # 5	DECSWL	Single Width Line
ESC # 6	DECDWL	Double Width Line
ESC # 8	BBNALN	Screen Alignment
ESC (args	SCS0	Select Character Set for G0
ESC) args	SCS1	Select Character Set for G1
ESC 7	DECSC	Save Cursor
ESC 8	DECRC	Restore Cursor
ESC : S	BBNSEC	Save EAROM Context
ESC : [	BBNPSH	Push Context
ESC :]	BBNPOP	Pop Context
ESC : args B	BBNSHB	Set Host Baudrate
ESC : args C	BBNSRP	Set Region Parameters
ESC : args D	BBNDAC	Display All Characters
ESC : args I	BBNRESP	Set Response Initiator
ESC : args L	BBNSFP	Set Font Parameters
ESC : args M	BBNMEM	Memory Management
ESC : args P	BBNPSG	Load PSG
ESC : args R	BBNREC	Restore EAROM Context
ESC : args c	BBNPNTR	Pointer
ESC : args d	BBNDRAW	Draw Line
ESC : args e	BBNSEM	Select Emulation Mode
ESC : args f	BBNFILL	Fill Rectangle
ESC : args m	BBNMOV	Move Graphically
ESC : args o	BBNSTO	Set Text Operation
ESC : args p	BBNPNT	Draw Point
ESC : args r	BBNRAST	Rastop
ESC : args v	BBNSDO	Set Drawing Operation
ESC <	VT52	Enter ANSI Mode
ESC =	DECKPAM	Enter Alternate Keypad Mode
ESC =	VT52	Enter Alternate Keypad Mode
ESC >	DECKPNM	Enter Numeric Keypad Mode
ESC >	VT52	Enter Numeric Keypad Mode
ESC A	VT52	Cursor UP
ESC B	VT52	Cursor Down
ESC C	VT52	Cursor Right
ESC D	IND	Index
ESC D	VT52	Cursor Left
ESC E	NEL	Next Line
ESC ENQ	4010	Enquiry
ESC F	VT52	Enter Graphics Mode

ESC FF	4010	Erase Display
ESC G	VT52	Exit Graphics Mode
ESC H	HTS	Horizontal Tabulation Set
ESC H	VT52	Cursor to Home
ESC I	VT52	Reverse Line Feed
ESC J	VT52	Erase to End of Screen
ESC K	VT52	Erase to End of Line
ESC M	RI	Reverse Index
ESC P	DCS	Device Control String
ESC P : E	BBNDOE	Download or Execute
ESC P : args L	BBNLFC	Load Font Character
ESC P : args P	BBNPRNT	Print
ESC P : args s	BBNDPD	Display Pixel Data
ESC SUB	4010	GIN Mode
ESC Y args	VT52	Direct Cursor Address
ESC Z	DECID	Identify Terminal
ESC [args @	ICH	Insert Character
ESC [args A	CUU	Cursor Up
ESC [args B	CUD	Cursor Down
ESC [args C	CUF	Cursor Forward
ESC [args D	CUB	Cursor Backward
ESC [args H	CUP	Cursor Position
ESC [args J	ED	Erase in Display
ESC [args K	EL	Erase in Line
ESC [args L	IL	Insert Line
ESC [args M	DL	Delete Line
ESC [args P	DCH	Delete Character
ESC [args R	CPR	Cursor Position Report
ESC [args c	DA	Device Attributes
ESC [args f	HVP	Horiz. and Vert. Position
ESC [args g	TBC	Tabulation Clear
ESC [args h	SM	Set Mode
ESC [args l	RM	Reset Mode
ESC [args m	SGR	Select Graphic Rendition
ESC [args n	DSR	Device Status Report
ESC [args q	DECLL	Load LEDs
ESC [args r	DECSTBM	Set Top and Bottom Margins
ESC [args x	DECREQTPARM	Request Terminal Parameters
ESC c	RIS	Reset to Initial State

THIS PAGE

INTENTIONALLY

LEFT BLANK

APPENDIX C

PACKED PIXEL DATA FORMAT

Some BitGraph terminal commands use the packed data format to achieve higher information density in the characters transmitted to the terminal. In this format, information is transmitted to the terminal in 16-bit quanta. When less than sixteen bits of information are required (as in the end of a scan line in the **Display Pixel Data** sequence), the information should be left-justified in a 16-bit field.

A maximum of three characters is needed to encode sixteen bits of information. The information is transmitted as a sign-magnitude number, with the sign information being carried in the last character. The first two characters use octal codes from 0100 to 0177 ('@' to 'DEL') to encode six bits of information for a total of twelve bits. If the high six bits of the number being encoded are zero, the first character may be omitted; similiarly, both characters may be omitted if all of the twelve high-order bits are zero.

The least significant four bits and the sign of the number are encoded in the last character, which is always present, and uses octal codes from 060 to 077 ('0' to '?') to encode +0 to +15, and octal codes from 040 to 057 (' ' to '/') to encode -0 to -15.

While formatting characters (CR, LF, TAB and FF) may be freely interspersed with the data characters for formatting, any other control code terminates the command that is interpreting the packed data. The control code is then interpreted normally.

The following are examples of 16-bit quantities and their resulting packed data representations.

16-bit quantity	number of chars in packed form	packed form
0	1	0
0x8000	3	`@0
0xC000	3	P@<blank>
0x7FFF	3	_<DEL>?
0xFFFF	1	!
0x000F	1	?
0x0333	2	s3

9 GLOSSARY

This brief Glossary provides definitions of some important technical terminology used throughout this manual.

ASCII: the American Standard Code for Information Interchange, which assigns a digital code to each of 96 printing characters and 32 control characters. The ASCII characters are 7 bits long and may have an additional parity bit for error detection.

Bitmapped graphics: an implementation technique in computer graphics where each picture element (pixel) on the screen is represented by one bit of memory in the terminal.

Command sequence: a special sequence of ASCII characters used to send special commands to the terminal. Also called terminal command sequence. Many of terminal command sequences begin with the ASCII Escape character (ESC, octal code 33), and are referred to as escape sequences. Following the ESC character may be a Control Sequence Introducer (CSI), which is itself a prefix indicating a "private" sequence follows. In the BitGraph the CSI is ESC : and in the VT100 the CSI is ESC [. In both cases the spaces following the ESC are not actually transmitted, but are used for clarity in the specifations below.

Cursor: a distinctive mark (such as a flashing square or underline) that indicates where the next character will be displayed.

Direct cursor address: a method of moving the cursor anywhere on the display to write the next group of data; see Cursor.

Firmware: the MC68000 program that controls the actions of the BitGraph terminal and resides in Read-Only Memory (ROM) in the terminal.

Parameter: a value appearing within a command sequence. Selective parameters are designated by Ps and numeric parameters by Pn. If left blank or specified as 0, parameters default to the function-dependent values specified for them in Chapter 4.

Scrolling: a method of displaying new material when the video display is full. To scroll by one line, a new line is displayed at the bottom, moving all the previous lines upward, and the top line is discarded.

INDEX

BBN Computer Corporation 33 Moulton Street, Cambridge, Massachusetts 02238