

4105 COMPUTER DISPLAY TERMINAL

*Please Check for
CHANGE INFORMATION
at the Rear of This Manual*

First Printing APR 1983
Revised FEB 1986

Tektronix[®]
COMMITTED TO EXCELLENCE

Copyright © 1983, 1985 by Tektronix, Inc., Beaverton, Oregon.
Printed in the United States of America. All rights reserved.
Contents of this publication may not be reproduced in any form
without permission of Tektronix, Inc.

This instrument, in whole or in part, may be protected by one or
more U.S. or foreign patents or patent applications. Information
provided on request by Tektronix, Inc., P.O. Box 500, Beaverton,
Oregon 97077.

TEKTRONIX is a registered trademark of Tektronix, Inc.



MANUAL REVISION STATUS

PRODUCT: 4105A Computer Display Terminal

This manual supports the following versions of this product: Firmware Version 1 and up. Current Version: 4.

REV DATE	DESCRIPTION
APR 1983	Original Issue
JUN 1983	Revised: pages iii, 1-2, 1-3, 2-1, 2-4, 2-5, 2-6, 3-2, 3-3, 3-8, 3-10, 3-11, 4-1, 4-6 through 4-9, 4-12, 4-14, 4-19, 5-1, 5-3, 5-5, 5-9, 5-10, 5-12, 5-14 through 5-17, 5-19, 5-22, 5-23, 5-24, 5-26 through 5-30, 5-32 through 5-35, 5-37 through 5-41, C-1 through C-4, D-2, D-3, D-4, and IDX-2.
SEP 1983	Rewritten to incorporate JUN 1983 revisions and add Version 2 enhancements.
APR 1985	Rewritten to support 4105A, including Version 4 enhancements.
JUN 1985	Revised to correct miscellaneous errors: pages v, 2-4, 2-14, 3-7, 3-14, 3-17, 3-22, 3-23, 3-25, 3-27, 3-28, 3-29, 3-33, 3-39, 3-41, 3-45, 3-46, 3-47, 4-12, 5-27, 5-32, 5-33, 5-44, 5-45, 5-46, 5-52, 5-62, 5-63, 5-65, 5-66, 5-72, 5-76, 5-78, B-2, C-4, C-5, C-7, IDX-1 Appendix added, pages: G-1, G-2
SEP 1985	Revised: pages 2-8, 4-35, 5-9.
JAN 1986	Revised: pages 3-25, 3-33, 3-38, 4-5, 4-35, 5-36, 5-51, A-14.
FEB 1986	Revised: pages 4-35, B-8.

RECEIVED
MAY 10 1964
U.S. DEPARTMENT OF THE INTERIOR
BUREAU OF LAND MANAGEMENT
DENVER, COLORADO

CONTENTS

Section 1	INTRODUCTION	Page
	Where to Look for Information	1-1
	The Terminal's Features	1-2
	A Brief Overview of the Terminal's Architecture	1-4
	Host Command Modes	1-6
Section 2	COMMUNICATIONS CONCEPTS	
	Establishing Host Communications	2-1
	Host Communications Commands	2-5
	Buffering and Handshaking	2-6
	Buffering and Handshaking Commands	2-8
	Using Host Syntax From the Keyboard	2-9
	Controlling Security	2-10
	Using Answerback	2-10
	Suppressing Echo	2-10
	Security Commands	2-10
	Reporting Terminal Status	2-11
	Requesting Reports	2-11
	Report Commands	2-12
	Controlling Copiers and Printers	2-14
	Controlling Color Copiers	2-15
	Controlling Monochrome Graphics Printers	2-15
	Controlling Dialog Area Copies	2-16
	Copier and Printer Commands	2-16
	Making Copies	2-18
	Copy Commands	2-18
	Port Configuration for Nonconventional Uses	2-19
Section 3	SCREEN EDITING CONCEPTS AND COMMANDS	
	Screen Editing Concepts	3-2
	The Dialog Area and the Dialog Area Buffer	3-2
	The Alphanumeric Cursor	3-3
	Scrolling the Display	3-4
	Creating Fixed Regions in the Dialog Buffer	3-4
	Controlling the Dialog Display and the Keyboard	3-7
	Restoring the Terminal to a Known State	3-10
	Screen Editing Modes	3-12
	ANSI Mode	3-12
	EDIT Mode	3-12
	VT52 Mode	3-14
	ANSI and VT52 Syntax	3-15
	Rules for Issuing ANSI and VT52 Commands	3-15
	Saving Command Settings	3-15
	More Information About Commands	3-15
	Command Description Format	3-16
	ANSI Command Descriptions	3-17
	VT52 Command Descriptions	3-45

Section 4	GRAPHICS CONCEPTS	Page
	Using Graphics Commands	4-2
	Displaying Dialog Between a Host and a User	4-3
	Display Areas	4-3
	Controlling the Dialog Area and Dialog Area Buffer	4-4
	Building User Prompts Into a Program: An Example	4-5
	Emulating 4010 Series Terminals	4-6
	Dialog Area Commands	4-6
	Understanding the Graphics Display and Graphics Memory	4-7
	Understanding the Graphics Display	4-7
	Graphics Memory	4-9
	Terminal Space	4-10
	Displaying Colors	4-12
	Using Color Indices	4-12
	Specifying Colors for the Color Map	4-12
	Changing the Color Map	4-14
	Color Commands	4-16
	Creating Images With Graphics Primitives	4-17
	Implicit Command Modes	4-17
	Vectors	4-18
	Markers	4-20
	Displaying Text in the Graphics Area	4-21
	Panels	4-23
	Creating Images With Pixel Operations	4-25
	4105 Pixel Dimensions	4-25
	Writing Into the Pixel Viewport	4-25
	Using Pixel Operations: An Example	4-28
	Pixel Commands	4-30
	Using 4010 GIN (Graphics Input)	4-31
	Using Macros	4-32
	Volatile and Nonvolatile Macros	4-32
	Memory Requirements for Macros	4-32
	Defining Macros	4-33
	Executing Macros	4-33
	Defining Key Macros	4-33
	Defining Macros From the Host: An Example	4-35
	Defining Macros From the Keyboard: An Example	4-36
	Defining Key Macros: An Example	4-39
	Deleting Macros	4-40
	Macro Commands	4-40
	Putting Together a Graphics Program	4-41
	How the Terminal's Memory Works	4-41

Section 5	4100-STYLE COMMANDS AND REPORTS	Page
	Parameter Types	5-2
	Host Parameters	5-2
	Setup Parameters	5-6
	Command Conventions	5-8
	A Sample Command Description	5-9
	About Omitting Parameters	5-10
	Saving Command Settings	5-10
	More Information About Commands	5-10
	4100-Style Command Descriptions	5-11
	Reports	5-79
	Report Parameters	5-79
	Complex Variations of Report Parameters	5-81
	The EOL String	5-81
	Report Descriptions	5-81
Section 6	PROGRAMMING EXAMPLES	
	Initialization Routine	6-2
	Command Parameter Encoding Subroutines	6-3
	Terminal Report Decoding Subroutines	6-7
	Low-Level I/O Support Subroutines	6-12
	Sample Program	6-13
Appendix A	CODE CHARTS AND KEYBOARD MACROS	
	ASCII/North American Character Set	A-2
	United Kingdom Character Set	A-4
	French Character Set	A-6
	Swedish Character Set	A-8
	Danish/Norwegian Character Set	A-10
	German Character Set	A-12
	Supplementary Character Set	A-14
	Rulings Character Set	A-15

Appendix B ERROR CODES

Introduction	B-1
Error Codes	B-1
Severity Levels	B-1
Error Codes	B-2
Error Codes for 4100-Style Commands	B-2
Error Codes for ANSI Commands	B-8

Appendix C COMMAND SUMMARY TABLES

4100-Style Commands	C-2
ANSI-Style Commands	C-8
VT52-Style Commands	C-14

Appendix D ENCODED INTEGER PARAMETERS

Appendix E TEKTRONIX COLOR STANDARD

Appendix F PREDEFINED FILL PATTERNS

Appendix G THE 4104 TERMINAL

GLOSSARY

INDEX

ILLUSTRATIONS

Figure	Description	Page
1-1	The Terminal's Programming Model	1-5
1-2	Command Syntax Modes	1-7
3-1	Dialog Area and Dialog Area Buffer	3-3
3-2	Fixed Regions in the Dialog Buffer	3-5
3-3	Cursor Movement in Origin Mode Absolute	3-6
3-4	Cursor Movement in Origin Mode Relative	3-6
3-5	Screen Editing Modes	3-13
3-6	A Typical ANSI/VT52 Command Description	3-16
4-1	The Dialog Area and the Dialog Area Buffer	4-4
4-2	How Colors Are Displayed in the Dialog Area	4-5
4-3	Magnified View of Pixels in a Line	4-7
4-4	How Colors Map to the Screen	4-8
4-5	Screen Pixels and Graphics Memory	4-9
4-6	Terminal Space and the Default Window	4-10
4-7	How Windows Map Terminal Space to the Screen	4-11
4-8	The HLS System Color Cone	4-13
4-9	The Effect of Changing the Color Map	4-15
4-10	Two Methods for Displaying a Line	4-18
4-11	Line Styles	4-19
4-12	Marker Types	4-20
4-13	Graphtext Characteristics	4-21
4-14	Examples of Panels	4-23
4-15	Writing Into the Pixel Viewport Using RASTER WRITE	4-28
4-16	Writing Into the Pixel Viewport Using RUNLENGTH WRITE	4-29
4-17	Encoding a Macro	4-35
5-1	How to Encode Integer Parameters	5-3
5-2	How to Encode XY-Coordinates	5-5
5-3	XY-Coordinate Parameter Syntax	5-5
5-4	A Typical 4100-Style Command Description	5-9
5-5	Creating a Panel With Multiple Boundaries	5-12
5-6	Packing Color Indices Using Three Bits Per Pixel	5-37
5-7	Character Path Settings	5-61
5-8	Graphtext Rotation Examples	5-62
5-9	Line Styles	5-67
5-10	Marker Types	5-67
5-11	4014 Line Styles	5-77

TABLES

Table	Description	Page
2-1	Host Communications Commands	2-5
2-2	Buffering and Handshaking Commands	2-8
2-3	Security Commands	2-10
2-4	Report Commands	2-13
2-5	Copier and Printer Commands	2-17
2-6	Copy Commands	2-18
2-7	RS-232-C Pin Connections	2-19
3-1	4100-Style Commands That Control Text Display	3-9
3-2	ANSI Commands That Control Text Display	3-9
3-3	ANSI Commands for Restoring Terminal Settings	3-11
3-4	Commands That Control the Numeric Keypad	3-14
3-5	SCS (SELECT CHARACTER SET) Values	3-28
3-6	Digit-Only Parameters for the SGR Command	3-30
3-7	Prefixed Parameters for the SGR Command	3-31
3-8	RM (RESET MODE) and SM (SET MODE) Command Parameters	3-35
3-9	Cursor Key Mode Codes	3-35
3-10	Cursor Key Mode Codes	3-39
3-11	Numeric Keypad Programming Codes	3-41
3-12	Alternate Keypad Programming Codes	3-46
4-1	Dialog Area Commands	4-6
4-2	Color Commands	4-16
4-3	Vector Commands	4-19
4-4	Marker Commands	4-20
4-5	Graphics Area Text Commands	4-22
4-6	Panel Commands	4-24
4-7	Pixel Commands	4-30
4-8	Macro Commands	4-40
5-1	ALU Modes	5-13
5-2	Effects of ENABLE DIALOG AREA	5-20
5-3	Special Inquiry Codes	5-39
5-4	Default Dialog Area Color Indices	5-51
5-5	Graphtext Character Rotation	5-62
5-6	Graphtext Sizes	5-63
5-7	Default Graphics Area Color Indices	5-73
5-8	Special Inquiry Codes	5-82
5-9	Terminal Status Character Bits	5-86
5-10	Implicit Command Mode Status	5-86
C-1	4100-Style Commands	C-2
C-2	ANSI-Style Commands	C-8
C-3	VT52-Style Commands	C-14

Section 1

INTRODUCTION

This manual contains the reference information needed to develop and maintain application software for the Tektronix 4105 Computer Display Terminal.

Two other manuals have been included with the terminal:

- *4105 Computer Display Terminal Operators Manual*, which includes a tutorial that shows the operator how to use the terminal's features, information on how to establish communication with a host computer and how to connect the terminal to a copier or printer, and an explanation of the terminal's self-test features.
- *4105 Computer Display Terminal Reference Guide*, which contains essential programmers reference material in a condensed form.

There is also a service manual available, which you can order through your Tektronix Field Office (see your Operators Manual for part numbers and ordering information).

WHERE TO LOOK FOR INFORMATION

The following brief discussion of each part of this manual may help direct you to the specific information you need. For an overall understanding of how to best use the terminal's programming capabilities, read Sections 1 and 2, the concepts portion of Section 3, and all of Section 4.

- Section 1, *Introduction*, contains introductory information about the terminal, including a discussion of the terminal's architecture.
- Section 2, *Communications Concepts*, discusses establishing communications with a host computer, using input and output buffering and handshaking protocols, using host syntax from the keyboard, controlling security, reporting terminal status, controlling copiers and printers, and using the host port for nonconventional purposes.

- Section 3, *Screen Editing Concepts and Commands*, discusses screen editing concepts and contains detailed command descriptions for the ANSI X3.64 and VT52 screen editing commands.
- Section 4, *Graphics Concepts*, discusses the concepts that you must understand to write a graphics application program for the terminal, and covers how to use the terminal's numerous graphics features, how to encode macros, and how the terminal uses memory.
- Section 5, *4100-Style Commands and Reports*, contains a discussion of parameter types and encoding methods, an alphabetically organized dictionary of all 4100-style commands (including graphics commands, communications commands, and macro-definition commands), and finally, a discussion of the format and encoding methods the terminal uses in sending reports to the host.
- Section 6, *Programming Examples*, contains FORTRAN programming examples of encoding and parsing routines — you may find these examples useful when developing host application programs.
- Appendices include:
 - Code charts for each character set and keyboard diagrams that show the macro numbers transmitted by each key on the keyboard
 - A list of error codes
 - Tables that summarize the terminal's 4100-style commands and screen editing commands
 - A table of integers already encoded for use by a host program
 - An illustration showing how hue, lightness, and saturation are used to specify colors in the HLS color coordinate system
 - A chart showing each of the fill patterns
 - A glossary
 - An index

INTRODUCTION

THE TERMINAL'S FEATURES

The terminal offers a wide range of features to support color graphics and text editing applications. Briefly, these features include:

Flicker-Free Display. The terminal incorporates a 60-Hz noninterlaced raster-scan color display for flicker-free viewing. The color display has 480 by 360 pixel resolution.

Full-Feature Keyboard. The terminal's low-profile detachable keyboard includes: uppercase and lowercase ASCII characters, BREAK and ERASE keys, a 14-key numeric keypad, four special function keys, eight programmable function keys, and a Joydisk. Most keys have N-key rollover, which permits the terminal to capture a fast typist's keyboard entries even when keystrokes overlap. An autorepeat feature lets you choose whether or not the keys repeat when they are held down for more than one-half second.

Tektronix Joydisk. The Joydisk — part of the 4105's keyboard — gives users an easy way to scroll text horizontally or vertically while screen editing. During GIN (a means of inputting graphics data), the Joydisk controls the crosshair cursor.

Programmable Keys. You can program most keys so that a single keystroke can invoke a sequence of characters or commands.

Selectable Color Palette. You can select from 64 distinct color mixtures. You can display graphics in up to eight colors simultaneously, with an additional eight colors for text display.

User Control of Color. The terminal's *Interactive Color Interface* provides an on-screen menu for modifying colors from the keyboard. You can see the colors change as you modify them; then you can keep the new colors, try others, or return to the ones you started with.

Independent Graphics and Text Display. The whole display screen can be used as a *graphics area* that displays the graphics images you create. You can also designate all or part of the screen as a *dialog area*, which is displayed in front of the graphics. You can use the dialog area for text editing or to display the dialog between the terminal and a host computer.

You can choose whether the dialog area has an opaque background (and hides the graphics behind it) or a transparent background that lets the graphics show through the dialog — or you can make the dialog area invisible, so that the graphics image is all that appears on the screen.

4100-Style Graphics. You can create, manipulate, and display complex graphics images through host or keyboard commands originally developed for the powerful Tektronix 4100-Series Graphics Terminals.

With the 4100-style commands, you can create line drawings or closed polygons, filling the polygons with solid colors or with a variety of patterns, and then add labels in different sizes, colors, and rotation angles. You can also create images by manipulating individual pixels in the display.

The graphics images are created and stored in a 4096 by 4096 coordinate space, and are scaled for display on the terminal's 480 by 360 pixel display. Because of the dimensions of its coordinate space, the terminal can display data files used by higher resolution Tektronix terminals.

Programs written to support a 4105 terminal are upwardly compatible with Tektronix 4100 Series desktop terminals and with the more powerful 4110 Series terminals.

4010-Style Graphics Input (GIN). The 4105 terminal offers the same capability for graphics input as the Tektronix 4010 Series terminals. You can use most GIN application programs written for 4010 Series terminals without modification. When using GIN, the terminal operator moves the GIN cursor about on the screen and presses a key to transmit the cursor coordinates to the host.

Adaptable Alphanumerics. The terminal can display two types of alphanumeric characters — alphatext and graphtext.

- *Alphatext.* The terminal uses alphatext in the dialog area either for screen editing applications or to display the dialog between the user and the host computer.
You can display alphatext in uppercase and lowercase characters with definable attributes, including character color, background color, underlining, and blinking.
- *Graphtext.* Graphtext is a special type of text for use in the graphics area. You can resize graphtext, rotate it, and write it in different directions.

There are eight alternate character sets available for displaying alphatext and graphtext. These include six international character sets (ASCII/North American, United Kingdom, French, Swedish, Danish/Norwegian, and German) and two special character sets that contain rulings characters and other useful symbols.

Versatile Command Sets. The terminal has three separate command sets:

- *4100-style command set* — These commands control Tektronix 4100-style graphics, communications settings, 4010-style graphics input (GIN), and terminal status reporting.
- *ANSI-style command set* — These commands control ANSI X3.64 screen editing, dialog display, and keyboard characteristics. A single command configures the terminal to run applications developed for VT100 terminals.
- *VT52-style command set* — These commands control VT52 style screen editing and permit the terminal to run applications developed for VT52 terminals.

Commands sent from the host computer use opcodes and encoded parameters; host syntax is used to make the best use of the terminal's memory and to speed communication. Commands sent from the keyboard use *Setup syntax*, which uses easy-to-remember command names and simple keywords or integers as parameters.

Terminal-Level Security. The terminal supports the VT100 answerback feature, which allows you to store a password-like message in the terminal's nonvolatile memory. The host can verify the answerback string against a list of authorized users and thus control the data and programs that it lets the terminal access.

Help and Status Facilities. The terminal's help and status facilities allow you to quickly and easily query the terminal for the syntax or status of most commands or terminal settings. You can ask for information about a single command or terminal setting, or you can ask for information about a group (*cluster*) of commands or settings.

Storable Macros. You can define and store macros — sequences of commands that you can call up from the terminal with a single keystroke or from the host with a single command. You can specify that macros are executed locally at the terminal or that they are transmitted to the host.

Nonvolatile Memory. You can save macros and certain terminal settings by storing them in *nonvolatile memory* — a part of the terminal's memory that is not erased even when the power is turned off. This means that you can configure the terminal for a particular application and then save the operating parameters. When you turn the power on again, the terminal automatically remembers and uses the saved settings.

Compatibility With Existing Software. The terminal can use a variety of graphics software from Tektronix and from other sources, and is compatible with several screen editing programs. Some software packages that are compatible with this terminal are:

- Tektronix Plot 10 Interactive Graphics Library
- Tektronix Plot 10 Graphical Kernel System
- Tektronix 4100P01 Direct Terminal Interface
- SASGRAPH, from the SAS Institute, Inc.
- DISSPLA, from ISSCO (Integrated Software Systems Corporation)
- Some editing programs designed for use with VT100 or VT52 terminals, such as EDT, VI, and EMACS
- Most existing programs written for Tektronix 4010 Series terminals

Standard RS-232 Communications. The terminal has one full-duplex serial RS-232-C port, which communicates with a host computer at data rates up to 38400 baud. You can set and save the communications settings that your terminal needs for communicating with a specific host. Then, once the communications settings have been saved, the terminal's configuration will be correct at power-up, and all you'll need to do is turn it on and use it.

Versatile Copy Commands. You can send a copy of the the terminal screen to a copier or printer with a simple keystroke or a command issued from the host or keyboard. When you make a copy of the display, you can copy the entire display or copy just the graphics area or dialog area. You can reverse black and white in copies, choosing whether the black background of a display prints or not. Host commands are available to make copies for either screen editing or graphics applications.

Color and Monochrome Copier Support. The terminal has a Centronics-style hardcopy port that is compatible with a variety of copiers and printers. You can make color or monochrome copies of both text and graphics. The terminal makes text or graphics color copies on the Tektronix 4691, 4692, and 4695 Color Graphics Copiers. It makes monochrome text or graphics copies on the Tektronix 4644 Dot Matrix Printer, the Hewlett-Packard Thinkjet, and other monochrome printers with Centronics-style interfaces and Epson-style graphics protocol. The terminal makes text copies on Centronics-style monochrome printers that don't offer graphics.

Data Logging. You can easily create a hard copy log of all data written to the screen. The Tektronix-private parameters added to the ANSI MEDIA COPY command turn this data logging on and off.

A BRIEF OVERVIEW OF THE TERMINAL'S ARCHITECTURE

The following discussion will give you an understanding of the terminal's architecture. Throughout this discussion, refer to Figure 1-1.

You can think of the terminal as two different terminals in one box, each having its own set of commands and controlling its own area of memory. These terminals would be:

- A *graphics display terminal* that is compatible with most Tektronix 4010 Series graphics application programs. This terminal also controls the Interactive Color Interface, which is used to adjust the terminal's color map of displayed colors from the keyboard.
- A *text entry and editing terminal* that is compatible with the American National Standards Institute (ANSI) Standard X3.64 and the International Organization for Standardization (ISO) Standard 6429. Because the terminal adheres to these standards, you can configure the terminal to run most popular screen editing programs.

When the terminal functions as a *graphics terminal*, it processes commands in this sequence (use Figure 1-1 to follow along):

1. The program sends 4100-style commands from the host.
2. The TEK Mode Interpreter interprets these commands.
3. The TEK Mode Interpreter invokes the appropriate graphics routines that direct the Graphics Controller Firmware.
4. The Graphics Controller Firmware determines how the graphics images will be written to the graphics memory.

NOTE

The TEK Mode Interpreter can write text to the dialog area in the Alphanumeric Memory, but cannot edit or otherwise manipulate the dialog area. This limited alphanumeric text capability is represented by the dashed line in Figure 1-1.

When the terminal functions as a *text entry and editing terminal*, it processes commands in this sequence (use Figure 1-1 to follow along):

1. The program sends ANSI or VT52 screen editing commands from the host.
2. The ANSI/VT52 Mode Interpreter interprets these commands.
3. The ANSI/VT52 Mode Interpreter invokes the appropriate screen editing routines from the Alphanumeric Controller Firmware.
4. The Alphanumeric Controller Firmware writes text to the Alphanumeric Memory.

NOTE

Because the Alphanumeric Controller is used by all the interpreters, commands sent by a graphics program could affect stored parameters used later in an editing program. This means that application programs should leave parameters at their initial values upon exit. Otherwise the terminal user may be required to enter several Setup commands to reestablish the appropriate parameters when switching applications.

Most graphics commands and many screen editing commands can be sent directly from the keyboard. As indicated in Figure 1-1, commands entered from the keyboard are processed by a Setup command interpreter, bypassing the ANSI/VT52 and TEK command interpreters.

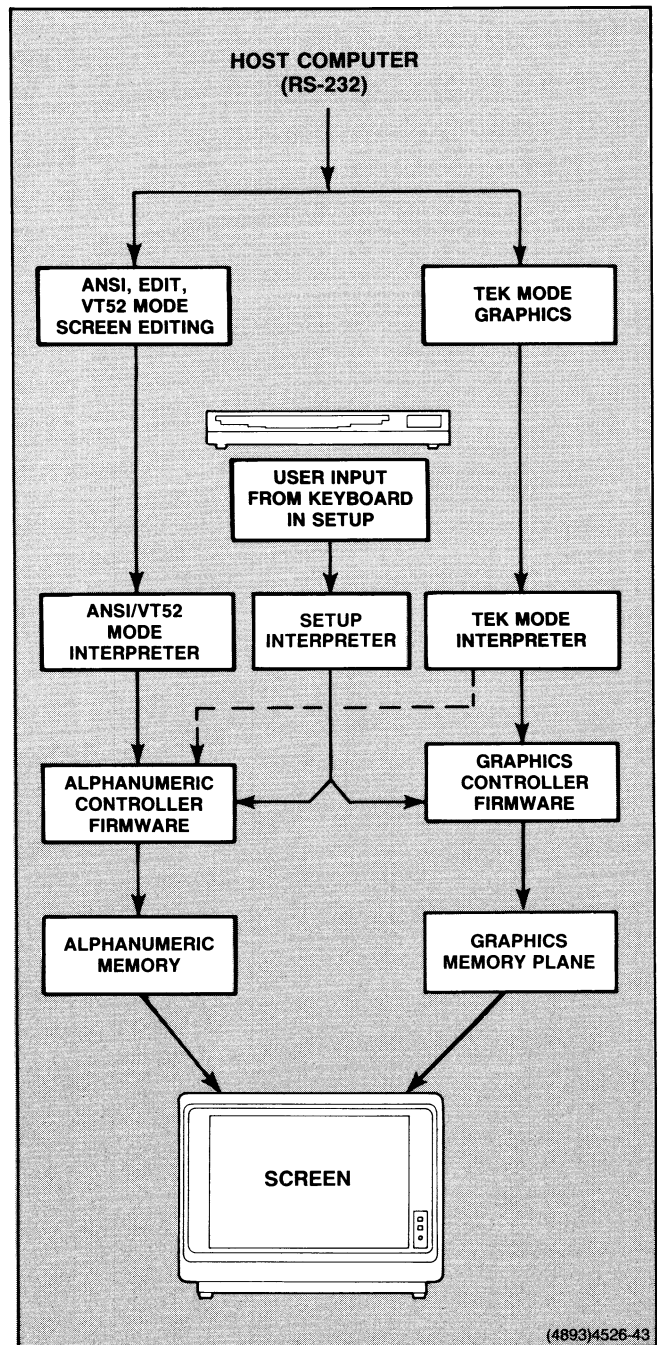


Figure 1-1. The Terminal's Programming Model.

HOST COMMAND MODES

As illustrated in Figure 1-2, the terminal has four modes for host operation: ANSI mode, EDIT mode, VT52 mode, and TEK mode. Typically, an application program running on a host computer selects the appropriate mode for its current task and sends commands to the terminal. In communicating with the host in any of these modes, the terminal accepts only the command syntax compatible with that mode.

Figure 1-2 shows these modes and lists the functional areas that each mode supports.

NOTE

The terminal's host command mode only affects host operations. When commands are entered from the keyboard, the terminal accepts any command that has Setup syntax.

Screen Editing

There are three screen editing modes that you can use to manipulate data in the terminal's Alphanumeric Memory.

- **ANSI mode.** In this mode, the terminal understands the syntax of the ANSI X3.64 commands only. Available functions are:
 - Screen editing — Includes commands to move the cursor; insert, delete, and erase lines; and report terminal status to the host.
 - Terminal control — Includes commands to enable or disable the keyboard; control the codes sent from the numeric keypad; and control display characteristics for the dialog area. The terminal control settings made in ANSI mode will carry over, as appropriate, into TEK mode and VT52 mode.

- **EDIT mode.** In this mode, the terminal understands the ANSI command set, but the operating characteristics of the terminal have been set to emulate a VT100 terminal.

EDIT mode automatically configures the terminal so that it is compatible with most VT100 software. While working in EDIT mode, you are actually still in ANSI mode and can use all the ANSI commands.

- **VT52 mode.** In this mode, the terminal is compatible with programs using VT52-style commands, and its operating characteristics are configured to emulate a VT52 terminal.

Section 3 of this manual contains a discussion of ANSI, EDIT, and VT52 modes, including their command syntax, command descriptions, and syntax examples.

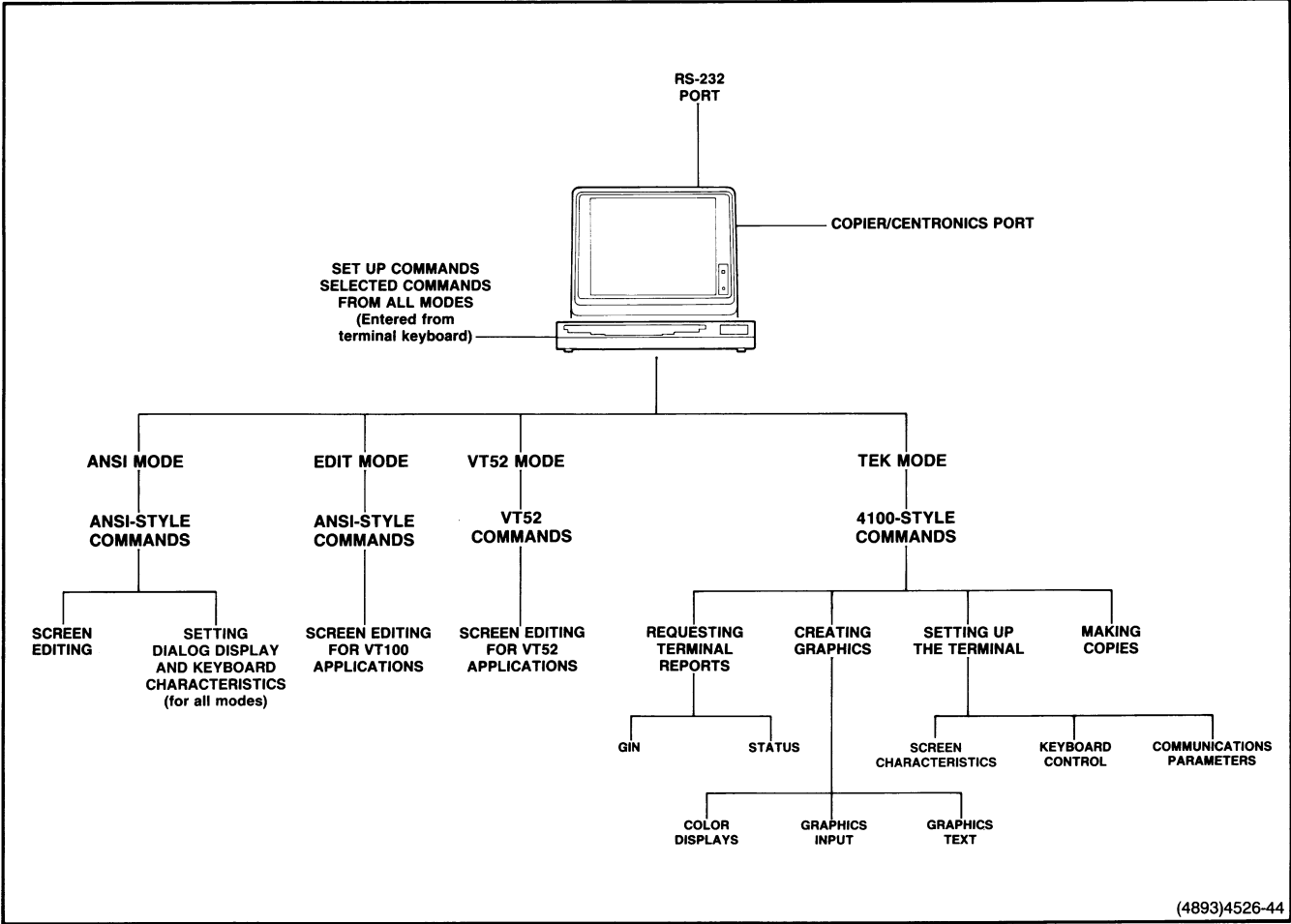
Graphics

In **TEK mode**, the terminal understands the syntax of the 4100-style graphics and terminal control commands. Some available functions are:

- Creating and controlling graphics images, graphics input (GIN), and graphtext
- Sending terminal status reports or GIN reports to the host
- Setting up the display, keyboard, and communications parameters for the terminal
- Making copies of alphanumeric and graphics data displayed on the terminal screen or stored on the host

Some terminal settings made in TEK mode (communications settings, for example) carry over, as appropriate, into ANSI mode — just as some settings made in ANSI mode are also effective in TEK mode.

Section 4 contains a discussion of graphics concepts, and Section 5 contains detailed descriptions of the commands available in this mode.



(4893)4526-44

Figure 1-2. Command Syntax Modes.

Section 2

COMMUNICATIONS CONCEPTS

This section discusses basic concepts for communications between your terminal and a host computer or a copier or printer. The topics covered are:

- *Establishing Host Communications*
- *Buffering and Handshaking*
- *Using Host Syntax From the Keyboard*
- *Controlling Security*
- *Reporting Terminal Status*
- *Controlling Copiers and Printers*
- *Making Copies*
- *Port Configuration for Nonconventional Uses*

Each discussion concludes with a table summarizing the commands that control the features described in that discussion.

The terminal's communications commands are a subset of Tektronix 4100-style commands. Take a look at the sidebar on this page for details about commands.

ESTABLISHING HOST COMMUNICATIONS

The terminal can communicate with a variety of host computers and modems through a full-duplex RS-232-C host port. You may need to change some of the default communications settings in order to establish communications. The appropriate values depend on your host computer or modem. Communications cannot take place unless these communications settings are correct; therefore, you'll need to use Setup syntax to change any settings; once communications have been established, you can issue commands from the host.

Whether set from the host or from the keyboard, the 4100-style commands in this section control communications in all host command modes (ANSI, EDIT, TEK, and VT52).

The host communications commands discussed in the following paragraphs are described in detail in individual command descriptions in Section 5. The table at the end of this discussion summarizes these commands.

COMMAND HINTS

- Commands are always identified in this manual by a descriptive name in uppercase letters.
- The terminal has three command sets and four host command modes. ANSI and EDIT modes use the ANSI command set; VT52 mode uses the VT52 command set; and TEK mode uses the 4100 command set.
- A program can freely switch command modes to access the terminal's full feature set (use the SELECT CODE command, which works in all modes).
- You can issue commands from a host program (using host syntax) or from the keyboard (using Setup syntax); the host and Setup versions of a command do exactly the same thing.
- When you select Setup (by pressing the Setup key), you can issue Setup commands from the keyboard without regard to the host command mode. Setup syntax uses simple keywords and ordinary integers.
- When you issue 4100-style commands from the host, the terminal must be in TEK mode and you must encode parameter values. The different parameter types (and their encoding schemes) are explained in Section 5. Section 6 contains sample routines for encoding these parameters.
- You can save many command settings in the terminal's nonvolatile memory by issuing the SAVE NONVOLATILE PARAMETERS command; then your terminal's settings will be appropriate for your application every time you turn on the terminal.
- 4100-style commands are described at the end of Section 5. ANSI and VT52 commands are described in Section 3. Each command is described in detail, listed alphabetically by command name.
- 4100-style reports are described at the end of Section 5. ANSI and VT52 reports are described alphabetically with the command descriptions in Section 3. Section 6 contains sample routines for decoding reports.
- If you're looking for information on a specific topic, try each section's detailed Table of Contents and functional listing of commands — you'll find these on each section divider. Also try the general Table of Contents and the Index.

COMMUNICATIONS

Baud Rate

The *baud rate* is the rate in bits per second that the terminal transmits or receives data. For example, *2400 baud* means that data is transferred at the rate of 2400 bits per second.

The terminal can transmit data to the host at a variety of internally controlled baud rates or be timed by an external clock.

Use the SET BAUD RATES command to set an appropriate baud rate. The appropriate value must be compatible with both the host computer and the communications line you are using.

You can use a single baud rate for both transmitting and receiving data, or you can specify split rates: one rate for transmitting and another rate for receiving.

The factory default baud rate for both transmitting and receiving data is 2400 baud.

Transmit Rate Limits

In some circumstances, the host computer may not be able to process information as fast as the terminal can send it.

You can use the SET TRANSMIT RATE LIMIT command to specify a maximum speed for terminal-to-host communications; the rate can be less than the baud rate specified by the SET BAUD RATES command. A transmit rate limit of 300, for instance, means that the terminal will space characters it sends to the host so that the average data rate is 300 bits per second.

Setting a transmit rate limit can be useful at high terminal-to-host baud rates, where the host computer cannot accept characters at the full data rate specified in the SET BAUD RATES command.

The factory default transmit rate limit is 19200 baud.

Stop Bits

While communicating with the host, directly or through a modem, the terminal sends and receives each character serially, as a sequence of ten or eleven bits. The first bit for each character is a *start bit*, always a 0. The next seven bits determine the particular ASCII character, after which comes a parity bit (described in the discussion which follows). The character ends with one or two *stop bits*, which are always 1. The communications line then remains in the marking condition until the start bit for the next character is detected. The terminal ignores the number of stop bits in characters coming from the host.

Use the SET STOP BITS command to specify the number of stop bits the terminal must use when it is transmitting data to the host.

The factory default is just one stop bit.

Parity

Data communications schemes sometimes use *parity bits* to indicate whether the sum of the data bits in a character is even or odd; this provides a simple test of the integrity of the transmitted data. The terminal's parity setting controls how the terminal sets the eighth bit (parity bit) in each character it sends to the host. (The terminal ignores the parity bit in data it receives from the host.)

Use the SET PARITY command to specify the parity scheme your host or modem requires.

The factory default sets the parity bit to 0 and uses no parity checking.

EOF String

This command defines the terminal's end-of-file string. When the terminal receives this string from the host during a COPY operation, it knows that the end of a file transfer has been reached and it terminates the COPY operation.

Use the SET EOF STRING command to set the EOF string that the terminal expects so that it matches the string that the host sends.

There is no factory default EOF string.

Break Time

Pressing the terminal's Break key sends a break signal to the host. In full-duplex communications, the break signal holds the communications line in a "space" condition. This is one way to interrupt host communications when the keyboard is locked.

Use the SET BREAK TIME command if you need to change the duration of the break signal, or to disable it altogether. Refer to your host's documentation to determine how the break signal is interpreted.

For hosts that do not accept break signals, set the break time to zero to disable the break feature.

The factory default break signal is 200 milliseconds, which is adequate for most host computers.

Echo

The characters that appear on the screen as you type are *echoes* of the typed characters. The echoes are usually transmitted from the host but may be provided by the terminal. If the host provides the echo, it transmits the characters it receives back to the terminal (this is *remote echo*). If the terminal is not communicating with the host or the host does not provide an echo, the terminal can provide its own echo (this is *local echo*).

You can enable or disable local echo with the SET ECHO command. If the host already provides an echo, disable local echo, or you will see double characters, LLIKKKEE TTHHISS, since both the terminal and host are echoing your entries. If the host or modem does not provide an echo, you should enable local echo so typed characters will appear on the screen.

The factory default is no local echo (the terminal won't display the characters unless the host echoes them).

Bypass Mode

Sometimes it is appropriate for the terminal to ignore characters sent from the host. For example, the terminal should ignore echoed characters that the host returns in response to a report; otherwise, the terminal might print the echoed characters as alphanumerics — or interpret them as xy parameters if the terminal is in Vector mode or Marker mode. Bypass mode provides a solution to this problem.

The terminal automatically enters Bypass mode before it sends most reports to the host. In Bypass mode, the terminal ignores all characters from the host until it receives the *bypass cancel character*. When the terminal receives this character, it exits Bypass mode and discards the bypass cancel character. Here's a typical sequence:

1. The host requests a report (by issuing a REPORT TERMINAL SETTINGS command, for example).
2. The terminal enters Bypass mode.
3. The terminal sends a report, terminated by the EOL string.
4. The host echoes the report to the terminal, but the terminal is in Bypass mode and ignores it.
5. The host sends the bypass cancel character).
6. The terminal cancels Bypass mode.
7. The host sends more data, which the terminal processes.

The terminal does *not* enter Bypass mode to send ANSI reports or the answerback string.

To ensure that the terminal exits Bypass mode at the end of each report, set the bypass cancel character to the last character that the host echoes when the terminal sends a report:

- If the host echoes $C_R L_F$ when it receives a C_R , set the bypass cancel character to L_F .
- If the host echoes only a C_R , set the bypass cancel character to C_R .
- If your host does not echo at all, you probably don't need Bypass mode, so set the bypass cancel character to N_U .

You can specify a different bypass cancel character by issuing the SET BYPASS CANCEL CHARACTER command.

There may be times when you want to force Bypass mode. For example, you might want to temporarily suppress the echo of a user's password during a login procedure. Issue the ENTER BYPASS MODE command before requesting the password, and issue a bypass cancel after it is received.

The factory default bypass cancel character is L_F (Line Feed).

D_T Filler Characters

Some host computers intersperse D_T characters (ADE 127) among the characters they send to a terminal. The host inserts these filler characters, and the applications program has no control over them. Since D_T is a valid character in integer and xy-coordinate parameters, these extra D_T characters can cause problems with Tektronix 4100-style commands.

The terminal includes two features that help solve this problem. First, it accepts the two-character sequence $E_C?$ as a synonym for D_T . Second, the IGNORE DELETES command causes the terminal to ignore any D_T characters from the host. The terminal does not, however, ignore $E_C?$ sequences.

Thus, if your host uses D_T as a filler character, take the following two steps:

- Change application routines that issue integer and xy-coordinate parameters to output $E_C?$ instead of D_T .
- Send an IGNORE DELETES command.

The factory default is for the terminal to accept D_T characters.

Host Communications Commands

Table 2-1 summarizes the commands used for setting up host communications through the host port. You can find detailed command descriptions in Section 5.

Hint. If you access a variety of computers that use different communications settings, you can store these communications setups as macros. This allows you to reconfigure the terminal simply by expanding the macro from the host or having the user press a programmed key. See the discussion *Using Macros* in Section 4.

Table 2-1
HOST COMMUNICATIONS COMMANDS

Descriptive Name	Setup Name	Function
IGNORE DELETES	IGNOREDEL	Determines whether the terminal ignores the <code>^D</code> (delete) character
SET BAUD RATES	BAUDRATE	Sets the terminal's transmit and receive baud rates
SET BREAK TIME	BREAKTIME	Sets the duration (in milliseconds) of the break signal the terminal sends when the Break key is pressed
SET BYPASS CANCEL CHARACTER	BYPASSCANCEL	Specifies the character that cancels Bypass mode
SET ECHO	ECHO	Specifies whether the terminal echoes characters it transmits to the host
SET EOF STRING	EOFSTRING	Specifies the terminal's end-of-file string
SET PARITY	PARITY	Specifies the kind of parity the terminal uses when it transmits data to the host
SET STOP BITS	STOPBITS	Specifies the number of stop bits appended to each character the terminal transmits
SET TRANSMIT RATE LIMIT	XMTLIMIT	Paces the terminal's data transmission so that it does not exceed the indicated rate

BUFFERING AND HANDSHAKING

Unless some provision is made for controlling communications, data can be lost. Often the terminal cannot process data immediately upon receiving it. Likewise, there may be times when the host cannot process data as fast as the terminal sends it.

For example, it takes longer to fill the inside of a polygon with a color or fill pattern than it does to receive and interpret the corresponding command from the host. Also, when the user puts the terminal in Setup, the terminal does not process information from the host until the user exits Setup.

The terminal tries to minimize data loss by *buffering* (storing transmitted data temporarily in the terminal's memory). Your program can help avoid data overflow in the input queue and output queue by using one of the following techniques:

- *Baud rate control* — The SET TRANSMIT RATE LIMIT command can slow down data transmission for selected applications, as discussed earlier in this section.
- *Handshaking* — Flagging schemes and Prompt mode prevent overflow of data in the input and output queues.

The buffering and handshaking commands discussed in the following paragraphs are described in detail in individual command descriptions in Section 5. The table at the end of this discussion summarizes these commands.

BUFFERING

The terminal uses its *input queue* and *output queue* to store data so that differences in the processing speed of the terminal and the host computer or other device do not cause data to be lost.

The Input Queue

The terminal's input queue is a part of program memory that holds incoming data until the terminal can process it. Data from the host is placed in the input queue; when the terminal is ready for that information, it removes the data from the queue and processes it (that is, executes commands, displays graphics, etc.). Then additional information can be stored in the queue.

For example, since the terminal cannot display characters while it is erasing the screen, it stores incoming data in its input queue until the screen is erased. Then it reads and processes the data from the queue.

As another example, the terminal can't process data from the host while it's in Setup, so it stores the data in the input queue, and then processes it when the user ends the Setup session.

You can set an appropriate input queue size by issuing the SET QUEUE SIZE command — you'll need to determine the best queue size for each application. A large queue ties up valuable program memory that can be better used to store graphics. On the other hand, a very small queue may overflow frequently, causing important data to be lost. You can use *flagging* to prevent data overflow in the input queue (flagging is discussed later in this section).

The factory default input queue size is 300 bytes.

The Output Queue

The terminal's output queue holds any characters that are waiting to be transmitted to the host. This includes any characters that have been typed on the keyboard (except in Setup or Local mode) and any reports that have been requested by the host.

When there are keyboard entries or reports waiting in the output queue, the terminal sends the data to the host one line at a time. If there are no characters backed up in the queue, then each character is sent as soon as it is typed (if the user types characters faster than they can be sent, some characters will be backed up in the queue).

HANDSHAKING

At high data rates, or for complex operations, input and output buffering may not be enough to prevent data loss. In this case, you should use *handshaking* to prevent the input queue from overflowing. Even at slow data rates, it may be wise to use handshaking.

You can use either one of two different handshaking schemes:

- *Flagging* — Prevents data overflow at either the host or the terminal
- *Prompt mode* — Prevents data overflow at the host

Flagging

Flagging is a handshaking scheme that prevents data overflow by allowing the terminal to start and stop data transmissions. If the host is capable of two-way handshaking, you can use flagging to prevent data overflow at both the terminal and the host. Two types of flagging are available: DC1/DC3 software flagging and DTR/CTS hardware flagging. You can use the SET FLAGGING MODE command to select the flagging scheme appropriate for your host.

The factory default is no flagging.

NOTE

If your computer assigns its hardware flagging signals to non-standard pins, you may be able to rewire the RS-232 connection to map the terminal's DTR/CTS signals to those required by the computer.

DC1/DC3 Flagging. DC1/DC3 flagging uses the DC1 and DC3 control characters (D_1 and D_3 , ADE 17 and 19, respectively) to enable or inhibit the transmitting device. DC1 is the start character and DC3 is the stop character. This allows you to use flagging with devices that do not control the RS-232-C DTR and CTS lines. You can use DC1/DC3 flagging for input, output, or both input and output.

DTR/CTS Flagging. In DTR/CTS flagging, the terminal indicates that it wants to transmit data by asserting DTR (Data Terminal Ready). If the host is ready to receive the data, it asserts CTS (Clear To Send). The terminal can transmit data to the host only when CTS is asserted.

If the terminal transmits characters faster than the host can process them, the host must drop CTS to stop transmission from the terminal. When the host is ready to receive more characters, it asserts CTS and the terminal resumes transmission.

When receiving characters from the host, the terminal uses the DTR signal line in the same way that the host uses the CTS line. If the host is sending characters faster than the terminal can process them, the terminal drops DTR, and the host stops transmitting to the terminal. When the terminal is ready for more characters, it asserts DTR, and the host resumes its transmission to the terminal.

NOTE

DTR/CTS flagging may not be practical when the terminal is connected to the host with a telephone line modem. In such circumstances, you should use DC1/DC3 flagging because the host does not have direct access to the DTR and CTS signal lines.

Prompt Mode

Prompt mode, which is controlled by the PROMPT MODE command, can prevent data overflow at the host. It allows the host to process each line of data before the terminal sends the next line of data.

A *line of data* has a specific meaning in this discussion. When data is typed on the keyboard, a line of data means “all the characters waiting to be transmitted, up to and including the next EOM character the user types (or the next EOL string the terminal sends).”

The EOM character is the character that the user types to end a command line to the host. The terminal’s factory default EOM character is **C_R**, but you can change it with the SET EOM CHARACTERS command.

Thus, if the terminal is set as it is when shipped from the factory, a *line of data* means “all characters waiting to be transmitted, up to and including the next **C_R** character.”

When the terminal is sending a report, a *line of data* means all the characters waiting to be transmitted, up to and including the next EOL string. The EOL (End of Line) string is explained in *Reporting Terminal Status*, later in this section.

Prompt mode uses a character string called the *prompt string* to manage terminal-to-host transmissions. You must use the SET PROMPT STRING command to set the prompt string to match the characters that a host issues when it is ready to receive more data.

Once you have enabled Prompt mode, it works like this:

1. The terminal has data in its output queue ready to send to the host. The data may be characters the user typed or reports from the terminal.

2. When the host is ready to receive data, it sends the prompt string to the terminal.
3. When the terminal receives the prompt string, it waits for a selected period of time (the *transmit delay*).

During the transmit delay, the terminal does nothing, unless it receives more characters.

- If the *host* transmits characters during the transmit delay, the terminal processes them, and waits for another prompt string.
 - If the *user* types characters during the transmit delay, they are added to the output queue. If the user’s entries fill the output queue, the terminal rings its bell and then ignores further entries until the end of the transmit delay.
4. After the transmit delay has elapsed, the terminal sends one line of data from the output queue to the host.
 5. The terminal repeats Steps 1, 2, 3, and 4 until Prompt mode is disabled, or until the output queue is empty.
 6. If the output queue is empty, the terminal sends characters to the host as fast as they are typed — until the user types the EOM character. The EOM character means that the terminal has sent a line of data to the host, so the terminal returns to the condition described in Step 1.

Buffering and Handshaking Commands

Table 2-2 summarizes the buffering and handshaking commands used to control the input queue. You can find detailed command descriptions in Section 5.

Table 2-2

BUFFERING AND HANDSHAKING COMMANDS

Descriptive Name	Setup Name	Function
PROMPT MODE	PROMPTMODE	Initiates or terminates Prompt mode
SET EOL STRING	EOLSTRING	Specifies the terminal’s end-of-line string
SET EOM CHARACTERS	EOMCHARS	Specifies the terminal’s end-of-message characters
SET FLAGGING MODE	FLAGGING	Specifies whether the terminal uses DC1/DC3 flagging or DTR/CTS flagging
SET PROMPT STRING	PROMPTSTRING	Specifies the string that initiates Prompt mode
SET QUEUE SIZE	QUEUESIZE	Specifies the size (in bytes) of the terminal’s input queue
SET TRANSMIT DELAY	XMTDELAY	Specifies the terminal’s delay (in milliseconds) between transmitting lines of text

USING HOST SYNTAX FROM THE KEYBOARD

The terminal allows you to use host syntax to issue commands from the terminal keyboard. This can be useful for debugging programs or testing the effect of commands that don't have Setup syntax. You can either use Setup or Local mode to issue commands.

In Setup, you can issue most 4100-style commands from the keyboard by typing the Escape character and the command's *opcode* (the two letters that identify the command) followed by any parameters needed for the command. You issue the parameters in their Setup form (that is, as simple integers or keywords), rather than using the encoded parameters required for commands issued from the host. You can't use this method to issue 4100-style commands that don't have opcodes — use Local mode instead.

In Local mode, you can issue ANSI, VT52, or 4100-style commands from the keyboard using host syntax (with 4100-style commands, this means that you need to encode integer and xy-coordinate parameters). To use Local mode, enter Setup (just press the Setup key), issue the LOCAL command (type *LOCAL YES*), and press the Setup key again to leave Setup and enter Local mode. In Local mode, the terminal:

- Processes the commands you issue from the keyboard, even though they are in host syntax.
- Provides a local echo, and displays as text anything the terminal doesn't recognize as a command.
- Stores any host input in its input queue, and processes it when you exit Local mode.

To resume communications with the host, you must reenter Setup and issue *LOCAL NO*.

In Local mode, the terminal doesn't display commands entered from the keyboard. To see host commands on the terminal screen, use Snoopy mode; in Snoopy mode, the terminal only displays the commands — it does not process them.

CONTROLLING SECURITY

The terminal allows you to protect secure data from being displayed on the screen. The *answerback* feature can control what data and programs are available to each terminal, while the SET ECHO and ENTER BYPASS MODE commands offer ways to temporarily suppress the display of characters entered at the keyboard.

USING ANSWERBACK

Answerback works like this: First, you must store a password-like message (called the *answerback string*) in the terminal's nonvolatile memory; then, when you want to allow access to host information only to authorized users, you can query the terminal for its answerback string. The host can then verify the string against a list of authorized users.

To define the terminal's answerback string, issue the SET ANSWERBACK STRING command, which must be issued from the keyboard. Be sure to follow it with a SAVE NONVOLATILE PARAMETERS command to save the string in nonvolatile memory. There is no factory default answerback string.

The host uses the ENQUIRY command to query for the answerback string. The ENQUIRY command is valid in all host command modes.

Suppressing the Answerback String

If the host provides the echo to the terminal screen, you may need to prevent the answerback string from being displayed. You can do this by entering Bypass mode just before issuing the ENQUIRY command. In this case, the answerback string must end with the EOL string, which takes the terminal out of Bypass mode (for details, see the discussion on Bypass mode earlier in this section). The next discussion, *Suppressing Echo*, describes other techniques for doing this.

SUPPRESSING ECHO

By temporarily suppressing echo, you can protect private or secure data entered from the keyboard — a login password, for example — from being displayed on the terminal screen. Depending on how the echo is provided, you may want to use the SET ECHO or the ENTER BYPASS MODE command.

- If the terminal provides the echo, use SET ECHO to temporarily turn off local echo.
- If the host provides the echo, turn it off at the host if you can. If you can't control the host echo, issue ENTER BYPASS MODE so that the terminal ignores all characters from the host.

SECURITY COMMANDS

Table 2-3 summarizes the commands you can use to provide security in your programs. You can find detailed command descriptions in Section 5.

Table 2-3

SECURITY COMMANDS

Descriptive Name	Setup Name	Function
ENQUIRY	(none)	Queries the terminal for its answerback string
ENTER BYPASS MODE	(none)	Causes the terminal to ignore data (echoes or reports) transmitted from the host
SET ANSWERBACK STRING	ANSWERBACK	Assigns the terminal's answerback string
SET ECHO	ECHO	Specifies whether or not the terminal echoes data entered at the keyboard

REPORTING TERMINAL STATUS

To control a 4105 terminal from a host, your host program must send commands to generate reports that the program can then parse. This discussion describes what reports are available to you and how to obtain them.

Each report has a unique format. The elements of 4100-style reports are encoded; the discussion *Reports* in Section 5 explains each report in detail and explains how to decode its elements. Each ANSI and VT52 report is described in detail in its own alphabetic entry in Section 3. Also take a look at the Section 3 and Section 5 command descriptions for the commands that generate reports.

NOTE

The host program must be written to decode 4100-style report parameters. The encoding scheme for each type of parameter and the syntax of each report is explained in detail in the last part of Section 5.

The terminal ends each report with an EOL string. The default EOL string is `CR`, although you can specify other characters with the SET EOL STRING command.

REQUESTING REPORTS

Your program must use the appropriate host command mode to request a specific report. In general, you use 4100-style commands to request general information about the terminal and its settings and ANSI or VT52 commands to request information that is specific to screen editing.

There are exceptions. You must request the Error Report in TEK mode, but it reports the most recent errors, regardless of the mode the terminal was in when the error was detected. You can issue the REPORT SYNTAX MODE command in any host command mode.

The following paragraphs describe the different reports and how they interrelate.

Reporting Terminal Status

When you use the ANSI mode DSR (DEVICE STATUS REPORT) command to request terminal status, the terminal sends back a Device Status Report with a parameter value of 0 to indicate that it is functioning correctly. (The DSR command is also used for reporting the alpha cursor position; see the discussion *Reporting Cursor Position* for more details.)

Reporting the Answerback String

You can issue the ENQUIRY command from the host to query the terminal for its answerback string. This capability is described earlier in this section in the discussion *Controlling Security*.

Reporting the Host Command Mode

Your application can issue the REPORT SYNTAX MODE command from any host command mode to cause the terminal to report which host command mode is currently in effect. The Syntax Report sent in response to the REPORT SYNTAX MODE command is just the same as the report that you would receive if you issued the the REPORT TERMINAL SETTINGS command asking for information on the current setting of the SELECT CODE command.

Reporting Terminal Settings

Use the 4100-style REPORT TERMINAL SETTINGS command to report the current settings for any 4100-style command. If you use a special inquiry code, the REPORT TERMINAL SETTINGS command reports on memory availability, terminal model number, or firmware version number:

Memory Availability. Use the special inquiry code `?M` to report the amount of memory currently available and the largest contiguous block of memory left. (The Setup command *STATUS MEMORYBLOCKS* sends the same information to the terminal screen.)

Terminal Model. Use the special inquiry code `?T` to report the terminal's model number. (The Setup command *STATUS TERMINAL* sends the same information to the terminal screen.)

Firmware Version. Use the special inquiry code `00` to report the terminal's firmware version number. (The Setup command *STATUS VERSION* sends the same information to the terminal screen.)

COMMUNICATIONS

Reporting Screen Editing Characteristics

The two ANSI mode commands DA (DEVICE ATTRIBUTES) and TEKID (IDENTIFY TERMINAL) have the same effect: they cause the terminal to report that its screen editing characteristics are like those of a VT100 terminal with the advanced video option. The VT52 command IDENTIFY causes the terminal to report that it has VT52 capabilities.

Reporting Errors

The terminal stores the eight most recent error codes, and you can use the 4100-style REPORT ERRORS command to send a report containing them to the host. Although the REPORT ERRORS command must be issued in Tek mode, it reports *all* errors, regardless of the mode the terminal was in when the errors were detected.

Error Codes. When the terminal receives erroneous commands, it sends error codes to the screen (in Setup) or to the host (during host operations) to help identify the problem so you can issue the command correctly.

The error codes the terminal issues are consistently structured so you can interpret them easily. The first four characters of the error code consist of a two-character opcode, the parameter number, and an error type. In Setup, the terminal displays each error code along with a message that explains the error code and identifies the severity of the error. Appendix B, *Error Codes*, contains more detailed explanations of each error code the terminal issues.

Reporting Copier Status

The 4100-style REPORT 4010 STATUS command reports whether a copier or printer is connected to the terminal and is powered up.

Reporting Cursor Position

In Tek mode, use the REPORT 4010 STATUS command to report the alpha cursor position (or the GIN cursor position if 4010 GIN is enabled). The cursor position is given in 4010 xy-coordinates.

In ANSI mode, use the ANSI command DSR (DEVICE STATUS REPORT) to ask for a Cursor Position Report. This reports the alphanumeric cursor position in row-and-column format.

REPORT COMMANDS

Table 2-4 lists the commands that a host program uses to request reports, as well as what mode the terminal must be in when you issue the command. You can find detailed command descriptions in Section 3 (for ANSI and VT52 mode commands) and Section 5 (for TEK mode commands).

Table 2-4
REPORT COMMANDS

Command	Host Command Mode	Report Function
DA (DEVICE ATTRIBUTES)	ANSI	Reports that the terminal is similar to a VT100 terminal with advanced video option
DSR (DEVICE STATUS REPORT)	ANSI	Reports either the alpha cursor position (row and column) or that the terminal is ready and functioning correctly
ENQUIRY	All modes	Queries the terminal for its answerback string
IDENTIFY	VT52	Reports that the terminal is similar to a VT52
REPORT ERRORS	TEK	Reports the eight most recently detected errors, their severity levels, and how many times each error was detected
REPORT SYNTAX MODE	All modes	Reports the current host command mode (ANSI, EDIT, TEK, or VT52)
REPORT TERMINAL SETTINGS	TEK	Reports the current parameter values for a specified command, the amount of program memory available, the model number, or the firmware version number
REPORT 4010 STATUS	TEK	Reports copier status and alpha cursor position (in xy-coordinates); reports just GIN cursor position if 4010 GIN is enabled
TEKID (IDENTIFY TERMINAL)	ANSI	Reports that the terminal is similar to a VT100 terminal with advanced video option

CONTROLLING COPIERS AND PRINTERS

The terminal can make copies of the graphics area and the dialog area on the following copiers and printers.

For color graphics and dialog copies:

- Tektronix 4691 Color Graphics Copier
- Tektronix 4692 Color Graphics Copier
- Tektronix 4695 Color Graphics Copier

For monochrome graphics and dialog copies:

- Tektronix 4644 Dot Matrix Printer
- Hewlett-Packard ThinkJet Printer
- Printers with a Centronics-style interface and Epson-style graphics protocol

For monochrome dialog copies:

- Printers that use a Centronics-style interface

The terminal needs to know which copier or printer it is connected to so it can send the appropriate control information and data. Use the SELECT HARDCOPY INTERFACE command to tell the terminal the copier type (and issue the SAVE NONVOLATILE PARAMETERS command so the terminal will remember the copier type from session to session).

You can control many aspects of the copying process. You can copy data directly from the host or copy images from your terminal screen. When you make copies from the terminal screen, you can copy just the dialog area, just the graphics area, or both at once (this is called a screen copy), and you can select either standard or reduced-size copies.

Each type of copier or printer has unique capabilities. The following paragraphs describe the 4100-style commands that you use to invoke these special abilities.

The SELECT HARDCOPY INTERFACE command and the copier and printer commands summarized in the following paragraphs are described in detail in individual command descriptions in Section 5.

CONTROLLING COLOR COPIERS

You can make either paper copies or transparencies when you are using a Tektronix 4691, 4692, or 4695 Color Graphics Copier, and you can control the image by issuing any of the following commands before you request the copy:

- Use the SET COPY SIZE command to select standard (8½x11") or half-size copies. On the 4695 Color Copier, you can control the copy size of both dialog area and graphics area copies; on a 4691 or 4692 Color Copier, you can control the size of dialog area copies only. The factory default is the standard size copy.
- On the 4691 and 4692 Color Copiers, use the SET IMAGE ORIENTATION command to choose either a horizontal or vertical orientation of the copy image. The factory default image orientation is horizontal (with the long axis of the image aligned with the long axis of the paper or film).
- On the 4692 Color Copier, use the SELECT COLOR HARDCOPY IMAGE DENSITY command to select either high-density or low-density copies (measured in dots per inch). High density copies are sharp and detailed; low density copies are faster. The factory default is high density copies.
- On the 4692 Color Copier, use the SET COLOR COPIER REPAINT command to set the number of times each copy image will be overprinted (this gives a more saturated image on color transparencies). By default, each image is painted just once.

Note that these commands affect copies made with the HARDCOPY and 4010 HARDCOPY commands and the S Copy and D Copy keys. Copies made with the COPY command are not affected.

You may want to use the SAVE NONVOLATILE PARAMETERS command to save any command settings you make; then the terminal will give you the same kinds of copies each time without your reissuing these commands.

CONTROLLING MONOCHROME GRAPHICS PRINTERS

The HARDCOPY command, 4010 HARDCOPY commands, and D Copy and S Copy keys will also make copies on monochrome graphics printers that have a Centronics-style parallel interface and Epson-style graphics protocol. The HARDCOPY command lets you choose either a negative image (white copies as black and vice versa) or a positive image. Although you cannot use the commands discussed in *Controlling Color Copiers*, there are two 4100-style commands that do affect monochrome printers.

Controlling Line Terminations

You can set the terminal to send either a Carriage Return or a Carriage Return/Line Feed combination at the end of each line it sends to your printer; however, both the printer and the terminal must be set up to use the same line ending. If your terminal sends the wrong line endings for the printer, it can cause one of two problems:

- If the printer expects a Carriage Return/Line Feed combination and the terminal sends just a Carriage Return, all your lines of text or graphics will print on the same line — resulting in one unreadable black line.
- If the printer expects just a Carriage Return and the terminal sends a Carriage Return/Line Feed combination, the copies you make will have an extra blank line following each line of characters (that is, single-spaced text will be double-spaced, and graphics will have an extra blank line after each printed line).

By default, the terminal sends a Carriage Return/Line Feed combination. If your printer does not accept the terminal's line endings, you can use the SET HARDCOPY MONOCHROME ATTRIBUTES command to change the terminal's default.

This command affects graphics copies made on monochrome graphics printers and dialog copies made on other Centronics-style monochrome printers.

Mapping Colors to Black and White

Although the terminal can use up to eight colors in the graphics area and eight more colors in the dialog area, monochrome printers can only print in black and white.

By default, the terminal prints all colors in black except the background (Index 0). This will present a readable image in many cases. When you will want more control of the monochrome image, you can use the MAP INDEX TO PRINT command to specify which color indices print and which will not print.

CONTROLLING DIALOG AREA COPIES

When making dialog area copies on color copiers or monochrome printers, you can use the SET DIALOG AREA HARDCOPY ATTRIBUTES command to control the starting point, the number of pages, and how Form Feeds (FF) are interpreted.

On the 4691, 4692, and 4695 Color Copiers, you can use the SET COPY SIZE command to select a smaller copy size. The standard (8½x11") copy size prints up to 80 characters per line, while the small copy size prints up to 132 characters per line without wrapping.

You can copy the dialog area in TEK mode using the HARDCOPY or 4010 HARDCOPY command. In ANSI or EDIT mode, you can use the MC (MEDIA COPY) command. See the following discussion on copy commands.

COPIER AND PRINTER COMMANDS

Table 2-5 summarizes the commands that control hard copy devices. You can find detailed command descriptions in Section 5.

Table 2-5
COPIER AND PRINTER COMMANDS

Descriptive Name	Setup Name	Function
MAP INDEX TO PRINT	HCMAP	Specifies which graphics color indices will print and which will not print when sent to a monochrome graphics printer
SELECT COLOR HARDCOPY IMAGE DENSITY	HCDENSITY	Selects either low-density, fast copies or high-density, slow copies (4692 only)
SELECT HARDCOPY INTERFACE	HCINTERFACE	Identifies the type of copier or printer that is connected to the terminal
SET COLOR COPIER REPAINT	HCREPAINT	Specifies the number of times the copier overprints each image — used for transparencies (4692 only)
SET COPY SIZE	HCSIZE	Selects a small image size for dialog or graphics copies (4695 only) or for dialog copies only (4691 and 4692)
SET DIALOG AREA HARDCOPY ATTRIBUTES	HCDATTRIBUTES	Sets attributes for a dialog area copy
SET HARDCOPY MONOCHROME ATTRIBUTES	HCMONOCHROME	Specifies the line termination (C _R or C _R L _F) that the terminal sends to a monochrome printer
SET IMAGE ORIENTATION	HCORIENT	Specifies whether the copy image is placed horizontally or vertically on the paper or film

MAKING COPIES

There are four host commands that send data to color copiers or monochrome printers connected to the terminal. The 4100-style COPY and HARDCOPY commands will send graphics data or dialog area data to a copier or printer. The 4010 HARDCOPY command (also a 4100-style command) will make screen copies only. The ANSI command MC (MEDIA COPY) will make dialog area copies only.

The 4100-style copy commands summarized in the following paragraphs are described in detail in individual command descriptions in Section 5. The ANSI command MC (MEDIA COPY) is described in detail in Section 3.

The 4100-style command COPY sends data straight from the host to the copier or printer without processing the data (the commands for controlling color and monochrome copies have no effect). The COPY command depends on the host program to send appropriate data and control information to the copier or printer.

The 4100-style command HARDCOPY, on the other hand, copies data displayed on the terminal screen. It makes either screen copies or dialog copies. The HARDCOPY command lets you choose either a negative image (white copies as black and vice versa) or a positive image.

The 4100-style command 4010 HARDCOPY copies the screen (graphics area and dialog area together).

The ANSI mode MC (MEDIA COPY) command offers an alternative way to send a simple dialog area copy — it works just like the HARDCOPY command. This command's greatest advantage in this terminal is the added Tek-private parameters that select *data logging*, a means of logging all text written to the dialog area by simultaneously sending it to a printer or copier.

One of the data logging options you can select toggles data logging on and off. You may want to program a key on the keyboard so that users can toggle data logging on and off easily (see the discussion *Defining a Key Macro: An Example* in Section 4).

There are a number of commands (discussed earlier in this section) that control the appearance of copies made with the HARDCOPY and 4010 HARDCOPY commands or with the D Copy and S Copy keys.

Copy Commands

Table 2-6 summarizes the commands that you can use to make copies on color copiers and monochrome printers.

Table 2-6
COPY COMMANDS

Descriptive Name	Setup Name	Function
COPY	COPY	Sends information directly from the host to a copier or printer
HARDCOPY	(none — use D Copy key or S Copy key)	Sends a copy of the screen (or just the dialog area) to a copier or printer
MC (MEDIA COPY ^a)	AUTOPRINT	Sends dialog to the printer at the same time as it is written to the screen (can also send a simple dialog copy)
4010 HARDCOPY	(none — use S Copy key)	Sends a copy of the screen to a copier or printer

^a This is an ANSI command, described in Section 3.

PORT CONFIGURATION FOR NONCONVENTIONAL USES

The following discussion is included in this manual for those users who want to connect the RS-232-C host port to different devices than the port was designed to accept. If you use the host port in conventional ways, the cable supplied with the terminal or peripheral will work without alteration.

The host port is configured to receive and transmit data from a host computer or other RS-232-C compatible data communications equipment (DCE), requiring only that you cable the devices together.

If you want to use the host port for nonconventional uses, it is important to realize that not all RS-232-C devices use the same pin assignments, and not all devices will operate when connected with a standard RS-232-C cable. A connection must have not only the right pins but the right information on the pins. To achieve the appropriate pin connections, you may need to use a *null-modem cable*, which crosses the connections to permit DCE devices to be cabled to other DCE devices or DTE devices to other DTE devices.

RS-232-C Pin Connections

Table 2-7 shows the pins used for the host RS-232 connector.

Table 2-7
RS-232-C PIN CONNECTIONS

Pin	Function	Used by Host Port?
1	FGND (Frame Ground)	Yes
2	RDATA (Data Received)	Yes
3	TDATA (Data Transmitted)	Yes
4	RTS (Request to Send)	Yes (Set to Mark ^a)
5	CTS (Clear to Send)	Yes (Input if DTR/CTS handshaking)
6	DSR (Dataset Ready)	No
7	SGND (Signal Ground)	Yes
8	DCD (Data Carrier Detect)	No
12	SDCD (Secondary Data Carrier Detect)	No
15	TCLK (Transmitted Data Clock)	Yes (If external clocking enabled by BAUDRATE 1)
17	RCLK (Received Data Clock)	Yes (If external clocking enabled by BAUDRATE 1)
19	SRTS (Secondary Request to Send)	Yes (Set to Mark ^a)
20	DTR (Data Terminal Ready)	Yes (Output if DTR/CTS handshaking; otherwise set to Mark ^a)

^a "Set to Mark" means that a + 12 volt signal is present at the indicated pin.

Section 3

SCREEN EDITING CONCEPTS AND COMMANDS

This section explains the terminal's screen editing features and describes three host command modes (ANSI, EDIT, and VT52) that give the terminal different screen editing capabilities. Detailed command descriptions for each ANSI and VT52 command follow the concepts discussion.

The topics covered in this section are:

- *Screen Editing Concepts.* Explains how the terminal's dialog area works and how ANSI commands control both the dialog area display and the terminal's keyboard functions. The concepts discussed here apply to all three screen editing modes.
- *Screen Editing Modes.* Discusses the terminal's three screen editing modes, how they relate to one another, and what functions each mode performs.
- *ANSI and VT52 Syntax.* Describes the formatting conventions used in this section and gives the syntax rules for the ANSI and VT52 commands.
- *ANSI Command Descriptions.* Describes each ANSI command and report in detail.
- *VT52 Command Descriptions.* Describes each VT52 command in detail.

COMMAND HINTS

- Commands are always identified in this manual by a descriptive name in uppercase letters.
- The terminal has three command sets and four host command modes. ANSI and EDIT modes use the ANSI command set; VT52 mode uses the VT52 command set; and TEK mode uses the 4100 command set.
- A program can freely switch command modes to access the terminal's full feature set (use the SELECT CODE command, which works in all modes).
- You can issue commands from a host program (using host syntax) or from the keyboard (using Setup syntax); the host and Setup versions of a command do exactly the same thing.
- When you select Setup (by pressing the Setup key), you can issue Setup commands from the keyboard without regard to the host command mode. Setup syntax uses simple keywords and ordinary integers.
- When you issue 4100-style commands from the host, the terminal must be in TEK mode and you must encode parameter values. The different parameter types (and their encoding schemes) are explained in Section 5. Section 6 contains sample routines for encoding these parameters.
- You can save many command settings in the terminal's nonvolatile memory by issuing the SAVE NONVOLATILE PARAMETERS command; then your terminal's settings will be appropriate for your application every time you turn on the terminal.
- 4100-style commands are described at the end of Section 5. ANSI and VT52 commands are described in Section 3. Each command is described in detail, listed alphabetically by command name.
- 4100-style reports are described at the end of Section 5. ANSI and VT52 reports are described alphabetically with the command descriptions in Section 3. Section 6 contains sample routines for decoding reports.
- If you're looking for information on a specific topic, try each section's detailed Table of Contents and functional listing of commands — you'll find these on each section divider. Also try the general Table of Contents and the Index.

SCREEN EDITING CONCEPTS

Screen editing programs allow you to view and edit a computer file as a whole rather than just line-by-line. You can use most screen editing programs with this terminal as long as the program is compatible with the ANSI X3.64 and ISO 6429 standards.

The documentation supplied with your editing program should define any deviation the program makes from these standards. If there are any deviations, you may not be able to use all of your program's features with this terminal, and your program may not be able to use all of this terminal's features.

If your editing program is compatible with VT52 terminals, use VT52 mode; if your editing program is compatible with VT100 terminals, use EDIT mode. These host command modes configure the terminal to run most VT52 or VT100 applications programs.

NOTE

Remember that ANSI, EDIT, and VT52 modes are host command modes — when commands are entered from the keyboard, the terminal accepts any command that has Setup syntax, regardless of the host command mode.

Many 4100-style commands control attributes that affect all modes. Those 4100-style commands that affect screen editing are included in the discussions in this section. They are described in detail in Section 5, *4100-Style Commands and Reports*.

If you write your own screen editing program, it must use the commands in the ways described in this section.

THE DIALOG AREA AND THE DIALOG AREA BUFFER

The *dialog area* is a definable area of the terminal screen that displays information from an area of program memory called the *dialog area buffer*. The dialog area can display text for screen editing programs, or it can display the dialog between the user and the host.

All screen editing and entry occur in the dialog area, and the ANSI, EDIT, and VT52 commands work only in the dialog area. Before executing any screen editing programs, you must make the dialog area visible with the 4100-style command SET DIALOG AREA VISIBILITY, or by pressing the Dialog key.

Use the 4100-style command SET DIALOG AREA LINES to set the number of lines (from 2 to 30) that are visible in the dialog area; the default dialog area is 30 lines. The dialog area is as wide as the screen (80 columns).

The *dialog area buffer* (or simply, *dialog buffer*) is the part of the terminal's program memory that is used to store information for display in the dialog area. The number of lines in the dialog buffer can be greater than or equal to the number of lines in the dialog area; the maximum size of the dialog buffer depends on the other features you are using.

The default dialog buffer is 49 lines; you can change the number of lines with the 4100-style command SET DIALOG AREA BUFFER SIZE.

You can choose a dialog buffer width of 132 columns instead of the normal 80 columns — use the ANSI command SM (SET MODE) to set Column mode. The dialog area will still display just 80 columns at once, but the user can use the Joydisk to scroll the text horizontally to display the additional columns.

Note the relationship between the dialog area and the dialog buffer in Figure 3-1. Because the dialog buffer has been defined larger than the dialog area, only part of the dialog buffer is visible on the screen.

THE ALPHANUMERIC CURSOR

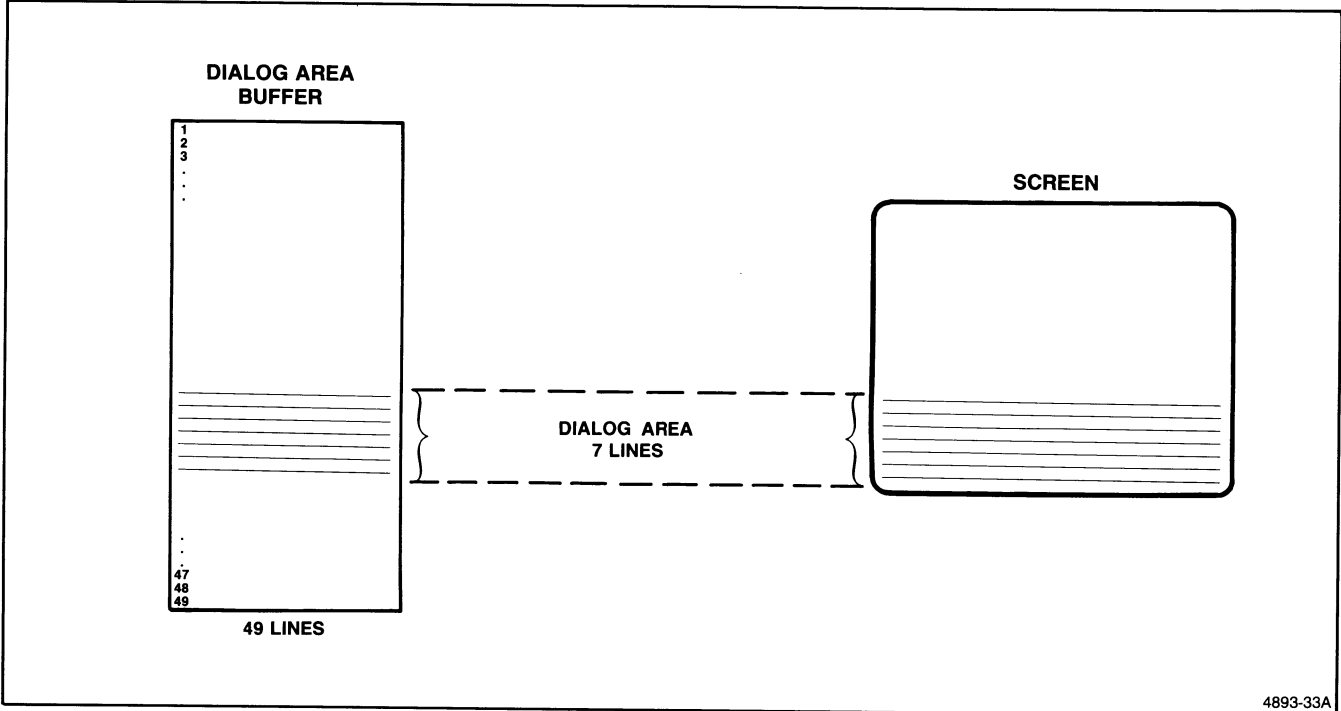
The alphanumeric cursor is an underscore displayed on the screen. The cursor serves as a pointer that indicates where commands should take place. For example, to delete a character, you first move the cursor to that character, then issue the command that deletes it.

The line of text on which the cursor is located is referred to in this manual as the *current line*. The position of the cursor on that line is referred to as the *current position*.

The terminal's screen is fully addressable. Each character on the screen has a unique horizontal row and vertical column address, and you can specify exact cursor positions using these row and column addresses.

To change the cursor color or make it blink, use the 4100-style command SET ALPHA CURSOR INDEX.

Your host program can move the cursor by issuing ANSI or VT52 cursor-movement commands. These cursor commands are listed on the next tabbed divider (preceding this section's command descriptions).



4893-33A

Figure 3-1. Dialog Area and Dialog Area Buffer.

SCROLLING THE DISPLAY

When the dialog buffer is larger than the dialog area, you can display the off-screen portions of the dialog buffer by scrolling. You can scroll the dialog buffer by issuing ANSI or VT52 cursor commands or by issuing ANSI scrolling commands.

Cursor Commands. ANSI and VT52 cursor commands let you scroll the dialog buffer vertically.

When (1) there are more lines of text in the dialog buffer than can be displayed in the dialog area, and (2) the cursor is visible on the screen, you can move the cursor to the top or bottom of the dialog area to scroll the undisplayed contents of the dialog buffer into view. (When the cursor is not displayed on the screen, cursor commands do not scroll the dialog buffer.)

Scrolling Commands. ANSI scrolling commands let you scroll the dialog buffer horizontally and vertically.

When you have selected a 132-column dialog buffer, you can use the ANSI commands SL (SCROLL LEFT) and SR (SCROLL RIGHT) to scroll the dialog area contents across the screen to display the additional columns.

When the dialog buffer contains more lines than the dialog area can display, you can use the ANSI commands SU (SCROLL UP) and SD (SCROLL DOWN) to scroll the dialog buffer contents up and down the screen to display the additional lines.

Note that the scrolling commands may move the cursor out of view.

The user can scroll the dialog buffer from the keyboard by pressing the Joydisk.

CREATING FIXED REGIONS IN THE DIALOG BUFFER

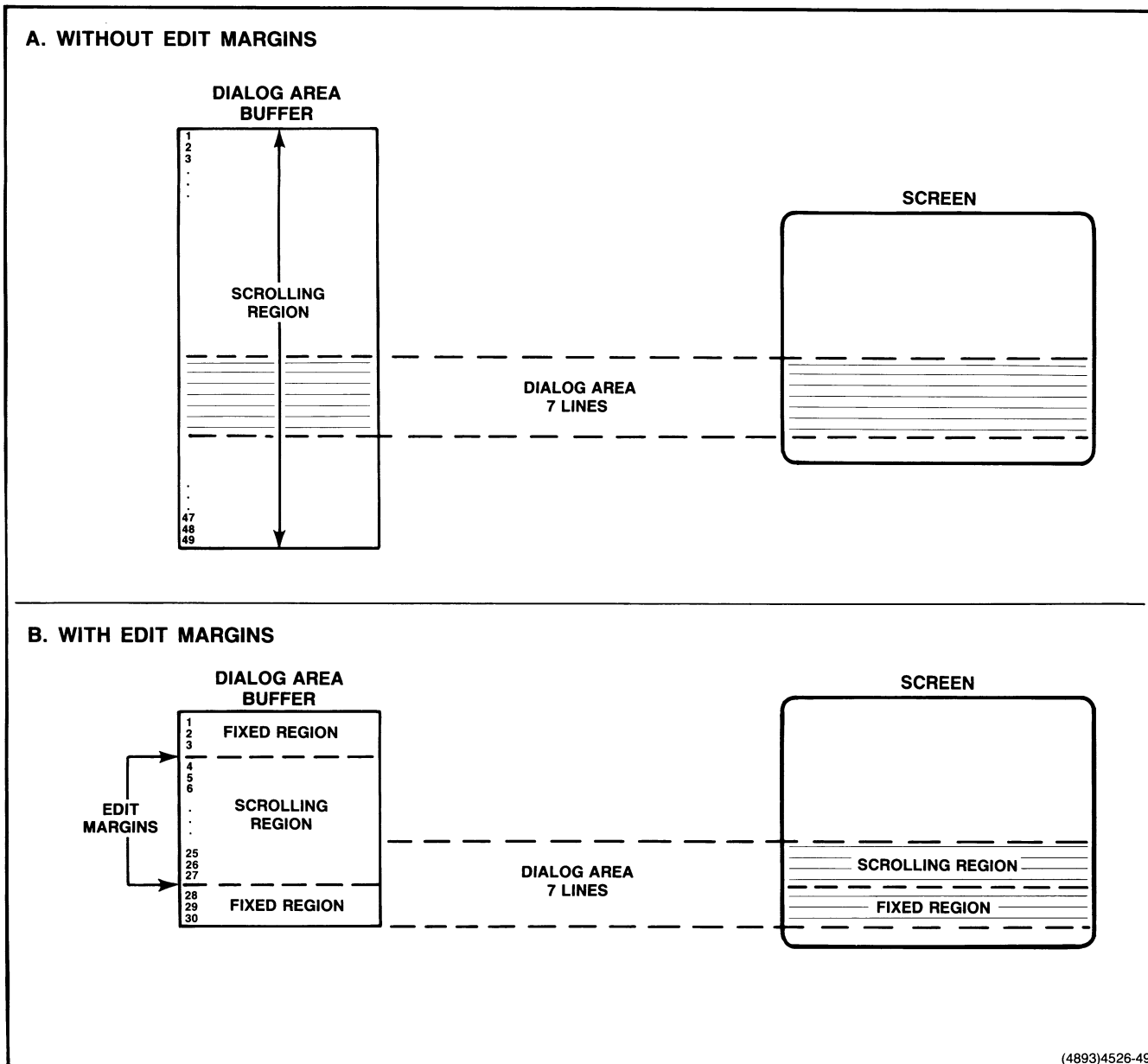
By default, the entire dialog buffer is treated as a *scrolling region*: when the buffer fills up, new lines of text are written to the bottom of the buffer, and the oldest lines of text (at the top of the dialog buffer) disappear.

You can choose instead to create one or two *fixed regions* on the screen. You might do this if your screen editing program has a status or message line at the top or bottom of the screen. Then you would use the scrolling region for editing and the fixed region for the status line.

Use the TEKSTBM (SET TOP AND BOTTOM MARGINS) command to define fixed regions. You specify top and bottom margins for the scrolling region; the lines between and including the top and bottom margins are in the scrolling region, and the lines outside the scrolling region are in the fixed regions. Although the TEKSTBM command is an ANSI command, the fixed and scrolling regions you create with it will remain in effect in EDIT mode and VT52 mode.

When you have defined one or more fixed regions, the dialog buffer is limited to 30 lines — the size of the terminal screen. When there are no fixed regions, the maximum size of the dialog buffer depends on the amount of program memory available (see *Managing Program Memory* in Section 4 for details).

Figure 3-2 shows the relationship between fixed and scrolling regions in the dialog buffer.



(4893)4526-49

Figure 3-2. Fixed Regions in the Dialog Buffer.

Addressing the Alphanumeric Cursor

When there are no fixed regions in the dialog buffer, cursor addresses represent buffer addresses (an address of Row 1, Column 1 means the first position in the buffer), and you can move the cursor anywhere in the buffer. However, when you have defined one or more fixed regions, you can choose whether cursor addresses represent addresses in the dialog buffer or addresses in the scrolling region within that buffer.

If you've set up fixed regions, you can use the ANSI commands SM (SET MODE) and RM (RESET MODE) to set *Origin mode*, which specifies the cursor addressing scheme. Origin mode also controls where the cursor can move within the fixed and scrolling regions of the dialog buffer.

Origin mode Absolute specifies that cursor addresses represent addresses in the dialog buffer. Origin mode Absolute lets you move the cursor to any character position in the dialog buffer — it is the only way you can move the cursor into a fixed region. Use the ANSI commands CUP (CURSOR POSITION) or HVP (HORIZONTAL AND VERTICAL POSITION) to move from one region to another.

Figure 3-3 shows the range of cursor movement with Origin mode Absolute.

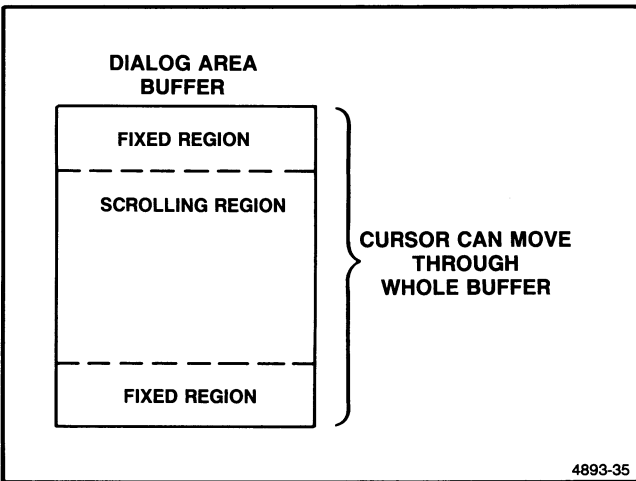


Figure 3-3. Cursor Movement in Origin Mode Absolute.

Origin mode Relative specifies that cursor addresses are based on the the first line of the scrolling region, as set by the TEKSTBM (SET TOP AND BOTTOM MARGINS) command. Origin mode Relative limits the cursor to the scrolling region.

For example, let's say you use the CUP (CURSOR POSITION) command and specify Row 1, Column 1. If Origin mode is Absolute, the cursor will move to Row 1, Column 1 in the *dialog buffer*; if Origin mode is Relative, the cursor will move to Row 1, Column 1 in the *scrolling region*.

Figure 3-4 shows the range of cursor movement with Origin mode Relative.

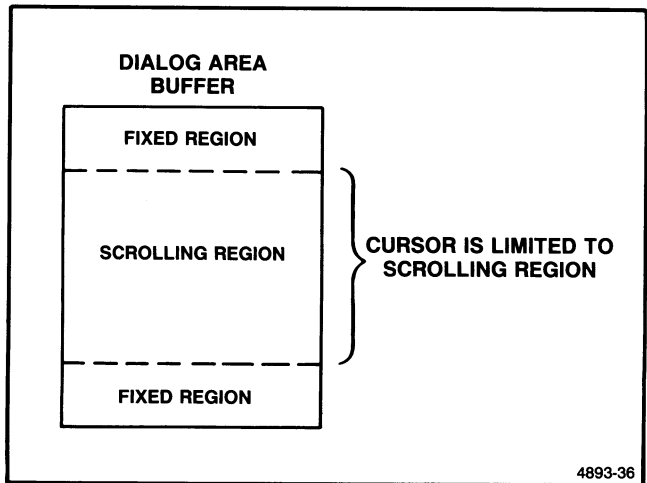


Figure 3-4. Cursor Movement in Origin Mode Relative.

CONTROLLING THE DIALOG DISPLAY AND THE KEYBOARD

Several ANSI commands control characteristics that affect the dialog display and keyboard features in all host command modes. These commands are:

- SCS (SELECT CHARACTER SET)
- SGR (SELECT GRAPHIC RENDITION)
- RM (RESET MODE)
- SM (SET MODE)

Although these are ANSI commands, they affect keyboard operation and dialog area display in all host command modes — ANSI, EDIT, TEK, and VT52 modes.

Selecting a Character Set

The characters displayed on your screen are taken from the default character set, which the terminal selects automatically when you plug in the keyboard. Eight character sets are stored in your terminal's firmware; plugging in the keyboard determines which one is displayed. For example, if you plug in the ASCII/North American keyboard, the terminal selects the ASCII/North American character set.

You may want to display characters from one of the international character sets or from the rulings or supplementary character sets instead. To do this, use the SCS (SELECT CHARACTER SET) command to designate that character set as the *G0 character set*.

You can also use the SCS command to specify both a *G0 character set* and a *G1 character set*. Then you can toggle between the two character sets — issue the $\$I$ (SHIFT IN) control character to invoke the *G0 character set* and the $\$O$ (SHIFT OUT) control character to invoke the *G1 character set*.

If you select a character set different from the one selected by your keyboard, the key caps will, of course, no longer correspond to the symbols displayed. Consult the ASCII or ISO code charts in Appendix A to find which keys on your keyboard generate the symbols you want.

Controlling Colors in the Dialog Area

You can control the colors available in the dialog area with either ANSI- or 4100-style commands. These commands control the colors assigned to the alphanumeric cursor, text characters, character-cell background, and the dialog area background.

The terminal uses the integers 0 through 7 as *color indices* to identify the colors. You use these indices to specify colors when you issue commands to change the colors in the dialog area. The default color assignments are:

0 = Black/Transparent	4 = Blue
1 = White	5 = Cyan
2 = Red	6 = Magenta
3 = Green	7 = Yellow

When Index 0 is used for text characters, it represents an opaque color just like Indices 1 through 7. However, when Index 0 is used for the character-cell background or the dialog area background, that background becomes transparent and whatever is behind it shows through. If both the character and dialog backgrounds are transparent, the dialog appears to be written on a piece of glass in front of the graphics area.

The 4100-style command SET DIALOG AREA COLOR MAP lets you reassign color mixtures to these color indices. (See the discussion *Defining Color Mixtures* in Section 4 to see how to specify colors.)

NOTE

The terminal operator can change the color mixtures in the dialog area color map from the keyboard by using the Interactive Color Interface. So if it is important to your program to have specific colors assigned to particular indices, you will need to reissue the SET DIALOG AREA COLOR MAP command to ensure that you get the colors you want.

You can control the color of the text and the dialog area background with either the ANSI command SGR (SET GRAPHIC RENDITION) or with the 4100-style command SET DIALOG AREA INDEX. Both commands offer the same control of color: you can specify the color of text characters, the color that fills the character cells around each character, and the color of the dialog area background.

You can use the SET ALPHA CURSOR INDICES 4100-style command to select one or two colors for the alphanumeric cursor. If you select two colors, the alpha cursor will alternate between them, appearing to blink.

Selecting Underscored or Blinking Text

You can specify underscored text or blinking text by issuing the ANSI command SGR (SELECT GRAPHIC RENDITION). When you turn on underscoring through the SGR command, all subsequent characters (including punctuation and spacing) will be underscored until you reissue the SGR command to turn off underscoring. You can turn blinking on and off in the same way. Once you have specified text as underscored or blinking, that text will retain the specified characteristic — turning off either characteristic affects only subsequent text.

Using the RM (RESET MODE) and SM (SET MODE) Commands

You can use the RM (RESET MODE) and SM (SET MODE) commands from a host application to control how the terminal writes text to the dialog area and how the keyboard functions. All but one of the modes you control with RM and SM can also be controlled from the keyboard with a Setup command.

These modes control the dialog display:

- *Autowrap mode (TEKAWM)* — Specifies whether characters in the rightmost column automatically wrap to the beginning of the next line.
- *Column mode (TEKCOLM)* — Selects an 80- or 132-column width for the dialog buffer.
- *Insert/Replace mode (IRM)* — Specifies whether characters are inserted into an existing line or write over existing characters.
- *Linefeed/Newline mode (LNM)* — Specifies whether a Line Feed sent to the dialog area also implies a Carriage Return.
- *Origin mode (TEKOM)* — Specifies whether cursor addresses are based on Row 1, Column 1 of the scrolling region or Row 1, Column 1 of the dialog buffer.
- *Overstrike/Replace mode (TEKORM)* — Controls how the Underscore and Space characters are treated when displaying formatted print files.
- *Screen mode (TEKSCNM)* — Reverses the colors on the display.
- *Send/Receive mode (SRM)* — Specifies whether or not the terminal echoes the data entered at the keyboard.

These modes control the keyboard:

- *Autorepeat mode (TEKARM)* — Specifies whether keyboard keys repeat when held down.
- *Cursor Key mode (TEKCKM)* — Specifies whether or not Function Keys F1 through F4 transmit ANSI cursor commands.
- *Keyboard Action mode (KAM)* — Locks and unlocks the keyboard.

Dialog Display and Keyboard Commands

Table 3-1 summarizes the 4100-style commands that control the dialog area and dialog buffer. These commands and are discussed in more detail in the command descriptions in Section 5. The discussion *Displaying a Dialog Between a Host and User* in Section 4, explains how to use the dialog area in graphics applications.

Table 3-2 summarizes the ANSI commands that your host program can issue to control the dialog area display and keyboard functions. If these commands are issued from the host, the terminal must be in ANSI mode. These commands affect display in all host command modes.

Table 3-1
4100-STYLE COMMANDS THAT CONTROL TEXT DISPLAY

Descriptive Name	Function
CLEAR DIALOG SCROLL	Erases the dialog buffer
ENABLE DIALOG AREA	Enables or disables the dialog area
SET DIALOG AREA BUFFER SIZE	Specifies the number of lines available for storing text in the dialog buffer
SET DIALOG AREA COLOR MAP	Specifies the color assigned to one or more color indices in the dialog area
SET DIALOG AREA INDEX	Specifies the color index for alphanum characters, character cell background, and dialog area background
SET DIALOG AREA LINES	Specifies the number of lines of the dialog buffer that are displayed on the screen
SET DIALOG AREA VISIBILITY	Makes the dialog area visible or invisible

Table 3-2
ANSI COMMANDS THAT CONTROL TEXT DISPLAY

Descriptive Name	Function
RM (RESET MODE)	Resets display characteristics or keyboard characteristics set by the SM (SET MODE) command
SCS (SELECT CHARACTER SET)	Selects a new character set (by designating it the G0 character set), or designates two character sets (as G0 and G1 character sets) so you can toggle between them
SGR (SELECT GRAPHIC RENDITION)	Sets display attributes for the dialog area: <ul style="list-style-type: none"> • Character Color • Character Cell Color • Dialog Area Background Color • Character Blink • Character Underscore
SI (SHIFT IN)	Invokes the G0 character set (designated by the SCS command)
SM (SET MODE)	Sets display characteristics or keyboard characteristics: <ul style="list-style-type: none"> • ANSI-to-VT52 mode • Autorepeat mode • Autowrap mode • Column mode • Cursor Keys mode • Insert/Replace mode • Keyboard Action mode • Linefeed/Newline mode • Origin mode • Overstrike/Replace mode • Screen mode • Send/Receive mode
SO (SHIFT OUT)	Invokes the G1 character set (designated by the the SCS command)

RESTORING THE TERMINAL TO A KNOWN STATE

ANSI and 4100-style commands offer a number of techniques to restore the terminal to a known state. These techniques allow you to reset the terminal to a predefined state or save current settings to be restored at a later time. Although you can reissue a string of commands every time you need to return to a known state, these techniques are far simpler.

For example, when your program starts, the terminal's editing and display characteristics may be in an unpredictable state because of commands issued by another program or by a user at the keyboard. You might want to reset the terminal to a predefined state specifically for screen editing applications.

Another example is when your program sends a status message to the terminal (or takes any action that interrupts the user's activities). You might need to save current settings so you can restore them when you are done.

The terminal's ANSI commands offer two simple ways to reset certain screen editing characteristics to a known state:

- You can use the ANSI command RIS (RESET TO INITIAL STATE) to reset a predefined set of characteristics to their original values. The RIS command:
 - Erases the screen
 - Moves the cursor to home (Row 1, Column 1 of the dialog buffer)
 - Sets Insert/Replace mode to Replace
 - Clears Edit Margins
 - Turns off text display characteristics (graphic rendition) set with the SGR (SELECT GRAPHIC RENDITION) command
 - Resets the G0 and G1 character sets to the default character set
 - Shifts in the G0 character set
 - Enables or disables the dialog area (depending on the saved setting for the 4100-style command ENABLE DIALOG AREA)
 - Makes the dialog area visible
- You can use the ANSI command TEKSC (SAVE CURSOR) to temporarily save the cursor position, the graphic rendition, and the character set currently in use. Then you can alter any of these settings temporarily — and when you want to return the terminal to its original state, you can use the ANSI command TEKRC (RESTORE CURSOR) to restore those saved values.

You can use various ANSI- and 4100-style commands to test the terminal's current settings and then reset individual characteristics. You can use the 4100-style command **REPORT TERMINAL SETTINGS** to report the current settings of a particular command. See the discussion on reports in Section 2.

The 4100-style command **FACTORY** returns the terminal to the default settings that it had when delivered from the factory. The 4100-style **RESET** command returns the terminal to its power-up settings (a combination of factory settings and any settings that have been saved in nonvolatile memory).

Table 3-3 summarizes the ANSI commands that the host program can issue to test and restore terminal characteristics.

Table 3-3
ANSI COMMANDS FOR RESTORING TERMINAL SETTINGS

Descriptive Name	Function
RIS (RESET TO INITIAL STATE)	Resets selected certain terminal attributes to their power-up values
TEKRC (RESTORE CURSOR)	Restores cursor position, graphics rendition, and character set previously saved with the TEKSC (SAVE CURSOR) command
TEKSC (SAVE CURSOR)	Saves cursor position, graphics rendition, character set, and Origin mode

SCREEN EDITING MODES

Figure 3-5 shows the terminal's screen editing modes, the functions that each supports, and how they relate to one another.

To use a screen editing program:

- The dialog area must be visible; use the 4100-style command SET DIALOG AREA VISIBILITY.
- The terminal must be in one of the screen editing modes; use the SELECT CODE command from any mode.

ANSI MODE

You can use ANSI commands to edit text or to set dialog display and keyboard characteristics (which affect the dialog display and keyboard functions in all host command modes).

ANSI editing commands allow you to:

- Move the cursor
- Delete characters and lines
- Erase characters and lines
- Insert characters and lines
- Create and delete tab stops and move the cursor to tab stops
- Choose double height, double width text display

The features of ANSI mode that affect the other screen editing modes are described in the preceding discussion *Controlling the Dialog Display and the Keyboard*.

You can find a complete list of ANSI commands, grouped by function, on the divider tab at the beginning of this section. Detailed descriptions of each ANSI command are given later in this section; see *ANSI Command Descriptions*.

EDIT MODE

EDIT mode is a submode of ANSI mode, and automatically configures the terminal so that it is compatible with most VT100 software. While working in EDIT mode, you are actually still in ANSI mode and can use all the ANSI commands.

Here's what the terminal does when it enters EDIT mode:

- Selects ANSI X3.64 syntax
- Enables the dialog area and makes it visible
- Sets the dialog buffer to 24 lines
- Defines the scrolling region to match the size of the dialog buffer (24 lines)
- Sets two ANSI modes:
 - Sets Origin mode to Absolute
 - Sets Insert/Replace mode to Replace (replaces rather than inserts characters)
- Temporarily disables the meanings associated with all user-programmed keys.

If you want user-programmed keys in EDIT mode, you can reenable their programmed meanings. Just switch temporarily to TEK mode to issue the 4100-style command ENABLE KEY EXPANSION — and then select ANSI mode so you can continue editing without disabling key expansion again.

If you select TEK mode from EDIT mode, the EDIT mode settings may not be appropriate, and you may need to reset them so your graphics program will work as expected. The SELECT CODE command description in Section 5 covers this transition in detail.

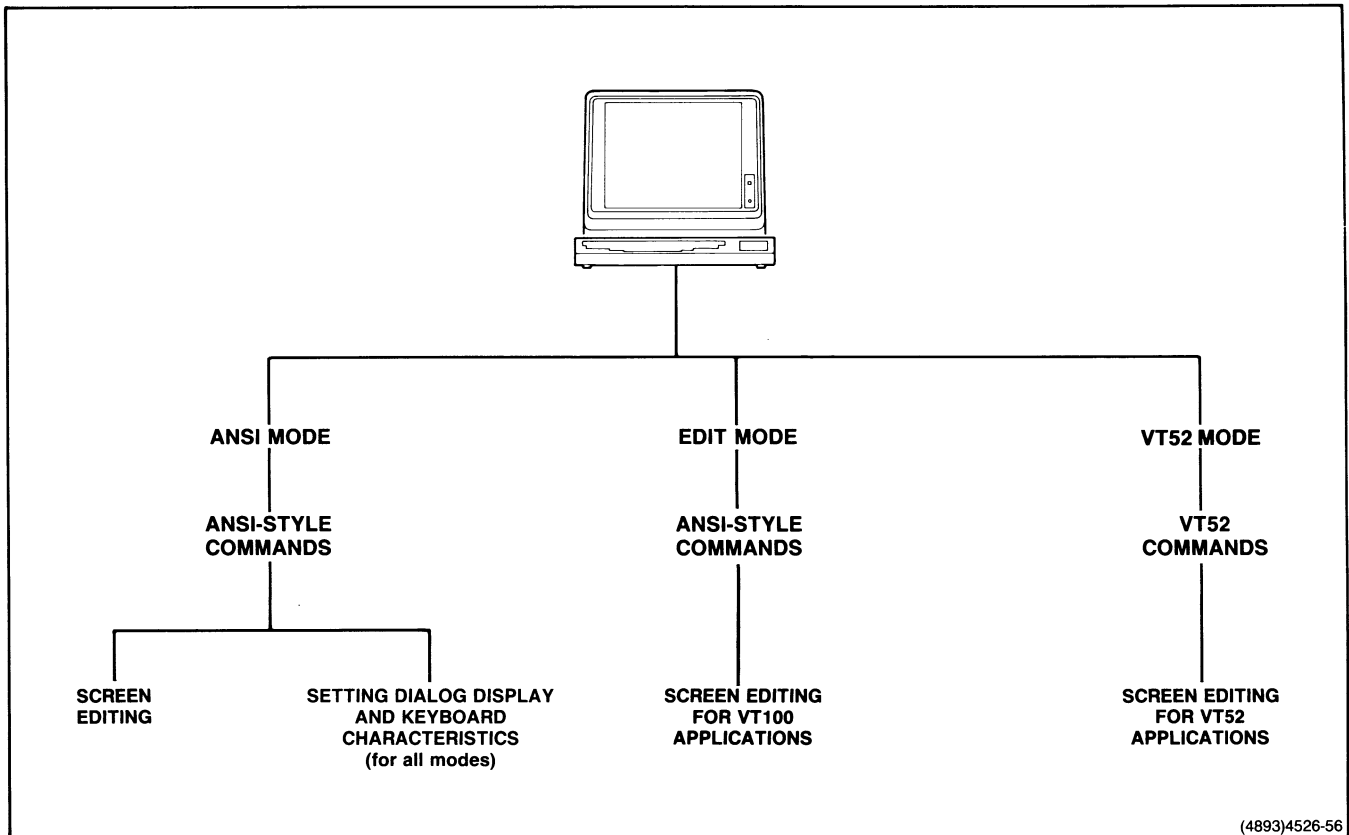


Figure 3-5. Screen Editing Modes.

VT52 MODE

VT52 mode configures the terminal to run most VT52 application programs. You can use only the VT52 commands, which are listed at the end of this section. A complete list of VT52 commands, grouped by function, is printed on the divider tab at the beginning of this section.

VT52 editing commands allow you to:

- Move the cursor
- Erase text
- Select host command modes
- Select VT52 modes
- Send reports

Although the ANSI commands that control the dialog display and keyboard are not available while in VT52 mode, you can issue them before entering VT52 mode and use the capabilities they give you while in VT52 mode. This means you can create fixed and scrolling regions in the dialog buffer and use any of the modes selected by the ANSI commands RM and SM.

VT52's Graphics mode provides another way to invoke the terminal's rulings character set. Entering Graphics mode selects the rulings characters as the G0 character set; exiting Graphics mode restores the character set to the G0 character set that was in effect prior to entering this mode.

VT52's Alternate Keypad mode gives the numeric keypad and Keys F5 through F8 special meanings; its effect is related to that of two ANSI-style commands:

- Issuing the VT52 command EXIT ALTERNATE KEYPAD MODE or the ANSI-style TEKKPNM command makes the keypad send *default codes*, while Keys F5 through F8 send special codes that are different in ANSI and VT52.
- Issuing the VT52 command ENTER ALTERNATE KEYPAD MODE or the ANSI-style TEKKPAM command makes both the keypad and Keys F5 through F8 send *alternate codes* while in VT52 mode, and *application codes* while in ANSI mode.

The keypad functions you select in one mode affects the keypad function in the other mode. If you switch back and forth between ANSI (or EDIT) mode and VT52 mode, you need to understand how the ANSI and VT52 keypad commands relate.

Table 3-4 shows how the commands that control ANSI and VT52 keypad modes interrelate. See Tables 3-11 and 3-12 later in this section for the specific values assigned to each keypad key and function key.

Table 3-4
COMMANDS THAT CONTROL THE NUMERIC KEYPAD

Descriptive Name	Mode	Function
ENTER ALTERNATE KEYPAD MODE	VT52	Selects alternate codes. When you select ANSI mode, you get ANSI application keypad codes.
EXIT ALTERNATE KEYPAD MODE	VT52	Selects default codes. When you switch to ANSI mode, you get the same keypad codes for all keys except F5 — F8.
TEKKPAM (KEYPAD APPLICATION MODE)	ANSI	Selects application codes. When you switch to VT52 mode, you get VT52 alternate keypad codes.
TEKKPNM (KEYPAD NUMERIC MODE)	ANSI	Selects default codes. When you switch to VT52 mode, you get the same keypad codes for all keys except F5 — F8.

ANSI AND VT52 SYNTAX

The ANSI and VT52 command descriptions in this section are consistently structured, using an easy-to-read set of syntax conventions. Figure 3-6 and the following discussion give a summary of the overall structure of command descriptions and of the notation used to show syntax.

RULES FOR ISSUING ANSI AND VT52 COMMANDS

Follow these rules when issuing ANSI and VT52 commands:

- In host syntax, issue the command as shown. An ANSI command may include the control sequence introducer (Esc), one or more parameters, and a command terminator character.
- Do not put separator spaces between parts of a command. (In a few cases, a Space character (SP) is a valid part of a command.)
- In host syntax, when a command has more than one parameter, separate them with semicolons.
- You can abbreviate the Setup name — just enter as many letters of the name as are needed to identify it uniquely. In the example in Figure 3-6, the Setup name *CODE* can be abbreviated *COD* (if you tried to abbreviate this to *CO*, the terminal would issue an error message since it wouldn't know whether you want to issue the *CODE* command or the *COLUMNMODE* command).
- In Setup syntax, enter parameters on the same line and separate them with a space or a comma.
- Most ANSI commands take integer values for their parameters. The widest valid range is 0 — 32767. If you specify a value higher than is reasonable for a particular parameter, the parameter defaults to the highest value that it can accept. You can omit leading zeros in ANSI commands issued from the host or in Setup.
- A few parameters for the MC (MEDIA COPY), RM (RESET MODE), SGR (SELECT GRAPHICS RENDITION), and SM (SET MODE) commands are Tektronix-private parameters. These parameters consist of a prefix (<, =, >, or ?) followed by an integer.
- The Setup versions of a few ANSI and VT52 commands use keyword parameters. These are simple words like *yes* or *insert*. You can abbreviate keyword parameters — you need to enter just enough of the keyword to make your choice clear. In Figure 3-6, where the keywords are *ANSI*, *EDIT*, *VT52*, and *TEK*, you could use just *A*, *E*, *V*, or *T* as parameter values.

SAVING COMMAND SETTINGS

You can save the settings of some commands by issuing the 4100-style command *SAVE NONVOLATILE PARAMETERS* before you turn off the terminal. Then the terminal will retain these settings in its nonvolatile memory even when it is powered off. The commands that you can save are identified following the command's statement of purpose with the phrase *Can be saved in nonvolatile memory*. You can find a list of all the commands that can be saved in nonvolatile memory in the *4105 Computer Display Terminal Programmers Reference Guide*.

MORE INFORMATION ABOUT COMMANDS

This manual and the *4105 Computer Display Terminal Reference Guide* provide summary information about the commands in several places:

- The divider tabs for this section list the ANSI and VT52 commands in functional groupings.
- Appendix C of this manual contains command summary tables. By scanning this appendix, you can, for example, see the parameter defaults for commands at a glance. You can find the Setup name of a command for which you know the descriptive name, and you can quickly find a command's host syntax and what its parameters are.
- The *4105 Computer Display Terminal Programmers Reference Guide* contains information from this manual, in summary form. It also contains additional cross-reference tables, which list commands by function and Setup name.

COMMAND DESCRIPTION FORMAT

Each command description is formatted in the following way:

- Command names are always shown in all uppercase characters at the beginning of the command description, followed by the command's function statement.
- The *Host Syntax* box shows the way a host application would send this command to a terminal.
- The *Setup Syntax* box shows the way you would enter this command at a terminal keyboard.
- The *Report Syntax* box shows the way the terminal reports information to the host.
- Characters shown in bold type are those that you must enter exactly as shown.
- Three periods (. . .) following a parameter name indicate that the command accepts multiple entries of the specified parameter.
- Default parameter values, if any, are shown at the end of each parameter description; when there is no default, the default value is shown as *(none)*.
- Many commands descriptions include syntax examples showing how to issue the command. When both host and Setup examples are included, the two examples do the same thing.

A Sample Command Description

Most ANSI and VT52 commands have only host syntax — they must be issued from the host while in ANSI or VT52 mode. A few commands, like the ANSI-style SELECT CODE command, shown in Figure 3-6, also have a Setup syntax. The figure uses the SELECT CODE command to illustrate this section's command description format:

- **Function Statement.** The command description begins with statement of the command's purpose following the command name.
- **Host Syntax.** This box shows how to issue this command from the host; send the escape sequence `Esc%!`, followed by the *syntax* parameter.
- **Setup Syntax.** This box shows how to issue this command from the keyboard; type the Setup name, **CODE**, followed by the *syntax* parameter and a carriage return.
- **Parameter.** There is a brief description of what the *syntax* parameter does. The description shows the valid values to use in the host version of the command and the keywords to use in Setup.
- **Default.** As the figure shows, the *syntax* parameter defaults to TEK mode.
- **Syntax Example.** Since the SELECT CODE command has both host and Setup syntax, the figure shows a typical command entry in each syntax.

SELECT CODE

Selects the host command mode, choosing ANSI, VT52, or TEK host command syntax. (Can be saved in nonvolatile memory.)

Host Syntax

```
Esc%! syntax
```

Setup Syntax

```
CODE syntax
```

syntax: specifies the host command mode that you want to use:

<u>Host</u>	<u>Setup</u>	
0	TEK	Selects TEK mode
1	ANSI	Selects ANSI mode
2	EDIT	Selects EDIT mode
3	VT52	Selects VT52 mode

Defaults: Factory = (none)
Omitted = TEK

This command causes the terminal to accept ANSI, EDIT, TEK, or VT52 commands from the host computer. The syntax of these different host command modes are not compatible. If you are using host commands from one mode and want to execute one or more commands from another mode, you must issue the SELECT CODE command with the appropriate parameter. This command is recognized in all host command modes.

Although this command does not affect the availability of commands entered from the keyboard in Setup, you can issue it while in Setup to cause the terminal to recognize a specific command syntax when it leaves Setup.

See the discussions *ANSI mode*, *EDIT mode*, and *VT52 mode* earlier in this Section for more information on the effect of this command on screen editing programs.

Selecting EDIT mode resets a number of terminal characteristics to emulate a VT100 terminal. When you select TEK mode after using EDIT mode, these terminal settings may not be appropriate, and you may need to reset them so your program will work as expected. See the command description for SELECT CODE in Section 5 for details.

Syntax Example

```
Host: Esc%!2  
Setup: CODE EDIT
```

Selects EDIT Mode.

4526-50

Figure 3-6. A Typical ANSI/VT52 Command Description.

ANSI COMMAND DESCRIPTIONS

This part of Section 3 contains descriptions of the terminal's ANSI commands. The commands are presented alphabetically according to their descriptive names.

BEL (BELL)

Sounds the terminal's bell.

Host Syntax

```
B_L
```

BS (BACK SPACE)

Moves the cursor left one position.

Host Syntax

```
B_S
```

The **B_S** character moves the cursor one character position to the left. If the cursor is already at Column 1, then **B_S** has no effect.

The Back Space key on your terminal's keyboard transmits the **B_S** character.

CAN (CANCEL)

Cancels an ANSI command in progress.

Host Syntax

```
C_N
```

When the terminal receives this character, it cancels any ANSI command currently being processed and inserts a **C_N** character at the current cursor position in the dialog area.

CBT (CURSOR BACKWARD TAB)

Moves the cursor backwards to a preceding tab stop on the current line.

Host Syntax

```
E_c[ number-of-preceding-tab-stops Z
```

number-of-preceding-tab-stops: specifies the number of tab positions the cursor moves to the left. A value of 1 moves the cursor to the preceding tab stop. A value greater than 1 (*n*) moves the cursor to the *n*th preceding tab stop on the current line.

Defaults: Factory = (none)
Omitted or 0 = 1

If you specify a number greater than the number of preceding tab stops, the cursor moves to Column 1 of the current line.

Syntax Example

```
E_c[3Z
```

Moves the cursor back three tab stops.

CHT (CURSOR HORIZONTAL TAB)

Moves the cursor forward to a following tab stop on the current line.

Host Syntax

```
E_c[ number-of-following-tab-stops I
```

number-of-following-tab-stops: specifies the number of tab stops the cursor moves to the right. A value of 1 moves the cursor to the next tab stop. A value greater than 1 (*n*) moves the cursor forward by *n* tab stops on the current line.

Defaults: Factory = (none)
Omitted or 0 = 1

If you specify a number greater than the number of following tab stops, the cursor moves to the end of the current line.

Syntax Example

```
E_c[3I
```

Moves the cursor forward three tab stops.

CPR (CURSOR POSITION REPORT)

Reports the row and column address of the current cursor position.

Report Syntax

```
Ec[ row ; column R
```

The Cursor Position Report is sent from the terminal to the host in response to a DSR (DEVICE STATUS REPORT) command with 6 as the parameter value.

If Origin mode is Relative (TEKOM set), the Cursor Position Report gives the row and column address in the scrolling region. In this case, Row 1, Column 1 is the upper-left corner of the scrolling region.

If Origin mode is Absolute (TEKOM reset), the Cursor Position Report gives the row and column address in the dialog buffer. In this case Row 1, Column 1 is the upper-left corner of the dialog buffer.

The terminal does not enter Bypass mode for the Cursor Position Report.

Syntax Example

```
Ec[22;55R
```

Reports that the cursor is at Row 22, Column 55.

CR (CARRIAGE RETURN)

Moves the cursor to the first column in the current line.

Host Syntax

```
CR
```

If the 4100-style command CRLF has been set so that C_R implies L_F, a line feed action is also performed.

The Return key on your terminal's keyboard transmits the C_R character.

CUB (CURSOR BACKWARD)

Moves the cursor left one or more columns.

Host Syntax

```
Ec[ number-of-columns D
```

number-of-columns: specifies the number of columns the cursor moves toward the left side of the screen. The cursor does not move beyond Column 1.

Defaults: Factory = (none)
Omitted or 0 = 1

If Column mode is set to 132, the cursor may disappear from the screen. This command will not scroll text horizontally to keep the cursor in view. Use the SL (SCROLL LEFT) or SR (SCROLL RIGHT) command or the Joydisk to scroll horizontally.

Syntax Example

```
Ec[10D
```

Moves the cursor back ten columns.

CUD (CURSOR DOWN)

Moves the cursor down one or more lines.

Host Syntax

```
Ec[ number-of-lines B
```

number of lines: specifies the number of lines the cursor moves toward the end of the dialog buffer.

Defaults: Factory = (none)
Omitted or 0 = 1

The cursor address is based on the first line of the dialog buffer (Row 1, Column 1 is the first position in the buffer), except when Origin mode is Relative and edit margins are set, in which case the cursor address is based on the first line of the scrolling region.

Syntax Example

```
Ec[5B
```

Moves the cursor down five lines.

CUF (CURSOR FORWARD)

Moves the cursor one or more columns to the right.

Host Syntax

```
Ec[ number-of-columns C
```

number-of-columns: specifies the number of columns the cursor moves toward the right side of the screen. The cursor does not move beyond the rightmost column.

Defaults: Factory = (none)
Omitted or 0 = 1

If Column mode is set to 132, the cursor may disappear from the screen. This command will not scroll text horizontally to keep the cursor in view. Use the SL (SCROLL LEFT) or SR (SCROLL RIGHT) commands or the Joydisk to scroll horizontally.

Syntax Example

```
Ec[5C
```

Moves the cursor five columns to the right.

CUP (CURSOR POSITION)

Moves the cursor to the specified row and column.

Host Syntax

```
Ec[ row-number ; column-number H
```

row-number: specifies the destination row for the cursor.

Defaults: Factory = (none)
Omitted or 0 = 1

column-number: specifies the destination column for the cursor.

Defaults: Factory = (none)
Omitted or 0 = 1

The cursor address is based on the first line of the dialog buffer (Row 1, Column 1 is the first position in the buffer), except when Origin mode is Relative and edit margins are set, in which case the cursor address is based on the first line of the scrolling region.

Syntax Example

```
Ec[5;12H
```

Moves the cursor to Row 5, Column 12.

CUU (CURSOR UP)

Moves the cursor upward one or more lines.

Host Syntax

```
Ec[ number-of-lines A
```

number-of-lines: specifies the number of lines the cursor moves toward the top of the screen.

Defaults: Factory = (none)
Omitted or 0 = 1

The cursor address is based on the first line of the dialog buffer (Row 1, Column 1 is the first position in the buffer) except when Origin mode is Relative and edit margins are set, in which case the cursor address is based on the first line of the scrolling region.

Syntax Example

```
Ec[20A
```

Moves the cursor up 20 columns.

DA (DEVICE ATTRIBUTES)

Tells the terminal to report what kind of terminal it is.

Host Syntax

```
Ec[0c
```

Report Syntax

```
Ec[?1;2c
```

The host sends this command with a parameter of 0 to the terminal asking it to identify what type of terminal it is. The terminal sends back to the host the report `Ec[?1;2c`, which says that the terminal is similar to a VT100 with Advanced Video Option. This means that the terminal includes:

- 132 Column mode
- Bold, blink, underline, and reverse image character attributes

If the host echoes this report back to the terminal, the terminal ignores the echo.

DCH (DELETE CHARACTER)

Deletes one or more characters beginning at the cursor position.

Host Syntax

```
Ec[ number-of-characters P
```

number-of-characters: specifies the number of characters to delete.

Defaults: Factory = (none)
Omitted or 0 = 1

Any characters to the right of the deleted characters are moved left by the same number of character positions; thus the gap is filled and the remainder of the line to the right of the last character is filled with spaces.

Only characters on the current line are affected by this command.

Syntax Example

```
Ec[10P
```

Deletes 10 characters starting from the current cursor position.

DL (DELETE LINE)

Deletes one or more lines starting with the current line.

Host Syntax

```
Ec[ number-of-lines M
```

number-of-lines: specifies the number of lines to delete.

Defaults: Factory = (none)
Omitted or 0 = 1

All lines following the deleted lines are shifted in a block toward the line containing the cursor. The cursor does not change position.

If you have defined fixed and scrolling regions, this command only affects lines in the region that contains the cursor. For example, if the cursor is in the top fixed region, only the lines in the top fixed region are affected.

Syntax Example

```
Ec[5M
```

Deletes five lines starting from the current cursor position — the lines below those that were deleted will be moved up.

DMI (DISABLE MANUAL INPUT)

Disables the keyboard.

Host Syntax

```
Ec \
```

Issuing this command is equivalent to using the ANSI command SM to set Keyboard Action Mode (KAM) or to issuing the 4100-style LOCK KEYBOARD command with a parameter of 1.

DSR (DEVICE STATUS REPORT)

Queries the terminal for a CPR (Cursor Position Report) or a DSR (Device Status Report).

Host Syntax

```
Ec[ status n
```

status: specifies which type of report you want. Valid values are:

- 5 Reports status in a Device Status Report
- 6 Reports cursor position in a Cursor Position Report

Defaults: Factory = (none)
Omitted = Error [n]l

When the host sends a DSR command with a parameter of 5, the terminal responds with a Device Status Report message with a parameter of 0 (that is, it responds `Ec[0n`, indicating that there is no malfunction).

When the host sends a DSR command with a parameter of 6, the terminal responds with a Cursor Position Report; see CPR (Cursor Position Report) for the syntax of the report.

If the terminal receives a DSR command with a parameter value of 0 (which could be the echo of a report it has sent to the host), the terminal ignores the command.

ECH (ERASE CHARACTER)

Erases one or more characters, starting at the cursor position.

Host Syntax

```
Ec[ number-of-characters X
```

number-of-characters: specifies the number of characters to erase.

Defaults: Factory = (none)
Omitted or 0 = 1

Characters are erased, not deleted. When a character is erased, its character cell is cleared (replaced with the current erase color index). The cursor location remains unchanged.

The effect of the ECH command is not confined to the current line. For example, if the cursor is in Column 41, and an ECH command with a parameter of 45 is issued, the 45 characters at and following the cursor position are erased. This is true even if this means erasing characters on following lines and into the fixed region from within the scrolling region.

Syntax Example

```
Ec[15X
```

Erases 15 characters starting from the current cursor position.

ED (ERASE IN DISPLAY)

Erases all or part of the dialog buffer.

Host Syntax

```
Ec[ erase-extent J
```

erase-extent: specifies the amount of text to erase:

- 0 Erases text from the cursor position to the end of the dialog buffer
- 1 Erases text from the beginning of the dialog buffer to the cursor position
- 2 Erases the entire dialog buffer

Defaults: Factory = (none)
Omitted = 0

The cursor does not change position.

Syntax Example

```
Ec[2J
```

Erases the entire dialog buffer.

EL (ERASE IN LINE)

Erases all or part of the current line.

Host Syntax

```
Ec[ erase-extent K
```

erase-extent: specifies the amount of text to erase:

- 0 Erases text from the cursor position to the end of the line
- 1 Erases text from the beginning of the line to the cursor position
- 2 Erases the entire line

Defaults: Factory = (none)
Omitted = 0

Syntax Example

```
Ec[0K
```

Erases from the current cursor position to the end of the line.

EMI (ENABLE MANUAL INPUT)

Enables the keyboard.

Host Syntax

```
Ecb
```

Issuing this command is equivalent to using the ANSI command RM to reset Keyboard Action Mode (KAM) or to issuing the 4100-style LOCK KEYBOARD command with a parameter of 0.

ENQUIRY

Queries the terminal for its answerback string.

Host Syntax

```
EQ
```

This command operates in any host command mode (ANSI, EDIT, VT52, or TEK mode). The terminal does not respond to this command in Local mode.

Your program can use the answerback string to identify the terminal and determine whether the terminal is authorized to use specific programs and data.

The terminal's answerback string can be set by using the setup command SET ANSWERBACK STRING, described in Section 5.

Note that, in TEK mode, the EQ character is a command terminator (like EC, FS, GS, and US).

FF (FORM FEED)

Indicates the start of a new page to a hardcopy unit.

Host Syntax

```
FF
```

This character inserts a FF character into the dialog area. (See command description for the 4100-style command SET DIALOG HARDCOPY ATTRIBUTES).

HT (HORIZONTAL TAB)

Advances the cursor to the next horizontal tab stop on the current line.

Host Syntax

```
HT
```

If there are no horizontal tab stops to the right of the cursor position, the cursor moves to the last column of the line.

When the terminal is shipped from the factory, there are tabs every eight columns, beginning in Column 1 (that is, in Columns 1, 9, 17, . . .). These tab stops can be changed and saved with the 4100-style SAVE NONVOLATILE PARAMETERS command.

The Tab key on your terminal's keyboard transmits the HT character.

HTS (HORIZONTAL TAB SET)

Sets a tab stop at the current cursor location.

Host Syntax

```
EcH
```

This ANSI command sets a tab at the current location of the alpha cursor. You can also use the 4100-style command SET TAB STOPS to set several tabs in a single command.

HVP (HORIZONTAL AND VERTICAL POSITION)

Moves the cursor to a specified row and column of dialog text.

Host Syntax

```
Ec[ row-number ; column-number f
```

row-number: specifies the destination row for the cursor.

Defaults: Factory = (none)
Omitted or 0 = 1

column-number: specifies the destination column for the cursor.

Defaults: Factory = (none)
Omitted or 0 = 1

If Origin mode is Relative (TEKOM set) and edit margins are set, the cursor address is based on the first line of the scrolling region (Row 1, Column 1 is the first position in the scrolling region). The cursor does not move beyond the top and bottom of the scrolling region.

If Origin mode is Absolute (TEKOM reset), the cursor address is based on the first line of the dialog buffer (Row 1, Column 1 is the upper-left corner of the buffer). The cursor may be moved beyond the top and bottom of the scrolling region by using the cursor positioning commands CUP and HVP.

Syntax Example

```
Ec[10;15f
```

Moves the cursor to Row 10, Column 15.

ICH (INSERT CHARACTER)

Inserts one or more Space characters at the cursor position.

Host Syntax

```
Ec[ number-of-characters @
```

number-of-characters: specifies the number of Space characters to insert.

Defaults: Factory = (none)
Omitted or 0 = 1

The character currently at the cursor position and all other characters to the right of the cursor are shifted *n* columns to the right. Characters shifted off the end of the line are lost. The cursor position remains unchanged.

Syntax Example

```
Ec[20@
```

Inserts 20 characters.

IL (INSERT LINE)

Inserts one or more blank lines in front of the current line.

Host Syntax

```
Ec[ number-of-lines L
```

number-of-lines: specifies the number of lines to insert.

Defaults: Factory = (none)
Omitted or 0 = 1

The current line and all following lines are shifted down, and lines scrolled below the bottom margin are lost. The cursor position does not change.

If fixed and scrolling regions have been defined, this command only affects lines in the region containing the cursor (if the cursor is in the scrolling (nonfixed) region, only the lines in the scrolling region are affected).

Syntax Example

```
Ec[5L
```

Inserts five blank lines.

IND (INDEX)

Moves the cursor position down one line without affecting the cursor position on the line.

Host Syntax

`ECD`

If the cursor is on the last line of the scrolling region, a blank line is added to the end of the scrolling region and a line is removed from the beginning of the scrolling region.

IRM (INSERT/REPLACE MODE)

Specifies whether each newly entered character replaces an existing character or is inserted at the cursor position.

Setup Syntax

`INSERTREPLACE mode`

mode: keyword; specifies whether characters are inserted or replace existing characters. Valid values are: *insert* and *replace*.

Defaults: Factory = replace
Omitted = replace

When Insert/Replace mode is *replace*, each character entered overwrites the character at the cursor position. When Insert/Replace mode is *insert*, each entered character is inserted at the cursor position and characters at and to the right of the cursor position move to the right.

The IRM command is part of the RM (RESET MODE) and SM (SET MODE) commands. See the description of the RM and SM commands to see how to change the settings of the IRM command from the host.

LF (LINE FEED)

Moves the cursor down one line.

Host Syntax

`LF`

If LNM (Linefeed/Newline mode) is reset (with the RM command), then `LF` has exactly the same effect as the IND (INDEX) command — it advances the cursor to the same position on the following line of text. If the cursor is on the last line of the scrolling region, a blank line is added to the end of the scrolling region and a line is removed from the beginning of the scrolling region.

If LNM (Linefeed/Newline mode) is set (with the SM command), then `LF` has the same effect as `CR` and IND combination: it advances the cursor position to the first character position on the following line.

LNM (LINEFEED/NEWLINE MODE)

Specifies whether a `LF` (Line Feed) character sent to the terminal also implies a `CR` (Carriage Return). (Can be saved in nonvolatile memory.)

Setup Syntax

`LFCR mode`

mode: keyword; specifies whether an `LF` character sent to the terminal also implies a `CR`. Valid values are: *no* and *yes*.

Defaults: Factory = no
Omitted = yes

This command has the same effect as the 4100-style LFCR command.

If Linefeed/Newline mode is selected, a `LF` (Line Feed) character sent to the terminal moves the cursor down one line without changing its column position. Otherwise, a `LF` also implies a `CR` (Carriage Return) character and so moves the cursor to the beginning of the next line.

The LNM command is part of the RM (RESET MODE) and SM (SET MODE) commands. See the description of the RM and SM commands to see how to change the settings of the LNM command from the host.

MC (MEDIA COPY)

Turns data logging on or off; can also be used by the host to make a hard copy of the dialog area.

Host Syntax

```
Ec[copy-option i
```

Setup Syntax

```
AUTOPRINT copy-option
```

copy-option: starts or stops transfer of data to a printer. Must be one of the following:

Host Setup

0	(none)	Copies the dialog area
?3	toggle	Turns data logging on or off
?4	no	Turns data logging off
?5	yes	Turns data logging on

Defaults: Factory = 0 (host), no (Setup)
Omitted = 0 (host), yes (Setup)

This command gives you two ways to copy the dialog area: the copy can be sent line-by-line as the dialog is created (this is called *data logging* or *autoprinting*), or you can issue the command from the host to make a simple dialog area copy.

NOTE

The data-logging feature does not work with the 4691 and 4692 Copiers (you can use the MEDIA COPY command to make a simple dialog copy on these copiers).

When you use the data-logging feature, the line endings sent to the dialog area control the line endings sent to the terminal. Lines terminated with a Line Feed (LF) or by the terminal's autowrap feature will be sent to the printer terminated with a Carriage Return (CR) followed by a LF. Lines that end with a Form Feed (FF) or Vertical Tab (VT) will be sent to the printer terminated with a CRFF or a CRVT, respectively.

In host syntax, you can make a simple dialog copy by issuing the MC command with a parameter value 0. This has the same effect as pressing the D Copy key or issuing the 4100-style command HARDCOPY with a parameter value of 3 (dialog area copy). Before making a dialog copy with Media Copy, you can issue the 4100-style command SET DIALOG AREA HARDCOPY ATTRIBUTES to control the position in the dialog buffer at which printing starts, the number of pages printed, and the way Form Feed characters are treated.

NOTE

If the first parameter value starts with a ?, then all subsequent parameter values are treated as if they began with a ?. This means that if you issue the MC command with parameter values 0 and ?5, the parameter value 0 must be issued first.

To comply with ANSI and ISO standards it is best not to mix standard parameter values (like 0) with Tektronix-private parameter values (like ?3 or ?4) in the same MEDIA COPY command. The terminal copies with MEDIA COPY commands that include both parameter types, provided the standard parameter value is issued first.

Syntax Example

```
Host: Ec[?3i
Setup: AUTOPRINT TOGGLE
```

Toggles data logging (turns it on if it's off, or off if it's on). See *Using Macros: Toggling the Data Logging Option* in Section 4 to find how to program a key to switch data logging on and off.

NEL (NEXT LINE)

Moves the cursor to the start of the next line.

Host Syntax

```
EcE
```

The NEL (NEXT LINE) command has the same effect as a CR and IND combination (a Carriage Return character followed by an IND command); that is, the cursor moves to the first character position on the next line.

REPORT SYNTAX MODE

Queries the terminal for a Terminal Settings Report giving the terminal's host command mode (ANSI, EDIT, TEK, or VT52).

Host Syntax

```
EC#!0
```

Report Syntax

```
%! mode
```

Issuing a REPORT SYNTAX MODE command has the same effect as issuing the 4100-style REPORT TERMINAL SETTINGS command for the SELECT CODE command (as if E_CIQ%! was sent from the host). See the discussion *Terminal Settings Report* in Section 5 for additional information.

The report you get back when you issue this command is in the format shown in the *Report Syntax* box. The *mode* report parameter identifies the host command mode currently in use at the terminal, and will always be one of the following:

- 0 = TEK mode
- 1 = ANSI mode
- 2 = EDIT mode
- 3 = VT52 mode

This command is recognized in all host command modes: ANSI, EDIT, TEK, and VT52.

RI (REVERSE INDEX)

Moves the cursor position up one line without affecting the cursor position on the line.

Host Syntax

```
ECM
```

If the cursor is on the first line of the scrolling region, a new line is added to the beginning of the scrolling region and a line is removed from the end of the scrolling region.

RIS (RESET TO INITIAL STATE)

Resets certain terminal attributes to power-up conditions (a combination of factory settings and any settings that have been saved in nonvolatile memory).

Host Syntax

```
ECC
```

When the terminal receives this command, it:

- Erases the screen
- Positions the alpha cursor at the Home position (Row 1, Column 1 of the dialog buffer)
- Sets Insert/Replace mode to Replace
- Clears edit margins
- Turns off the graphic rendition (text characteristics) set with the SGR command
- Selects the default G0 and G1 character set
- Shifts in the G0 character set
- Enables or disables the dialog area (depending on the saved setting for the 4100-style command ENABLE DIALOG AREA)
- Makes the dialog area visible

RM (RESET MODE)

Resets one or more terminal modes set with the SM (SET MODE) command.

Host Syntax

```
Ec[ mode . . . I
```

Setup Syntax

(See Table 3-8)

mode: resets one or more of the ANSI modes listed below. Most of these modes can be invoked by their own Setup command, shown in Table 3-8 (under the SM command description); you can also look each mode up separately under its mode name. Valid values are:

- 2 KAM (Keyboard Action Mode)
- 4 IRM (Insert/Replace Mode)
- 12 SRM (Send/Receive Mode)
- 20 LNM (Linefeed/Newline Mode)
- <1 TEKORM (Overstrike/Replace Mode)
- ?1 TEKCKM (Cursor Keys Mode)
- ?2 TEKANM (ANSI-to-VT52 Mode)
- ?3 TEKCOLM (Column Mode)
- ?5 TEKSCNM (Screen Mode)
- ?6 TEKOM (Origin Mode)
- ?7 TEKAWM (Autowrap Mode)
- ?8 TEKARM (Autorepeat Mode)

Defaults: Factory = (none)
 Omitted = Error [111]

The three periods (. . .) in the host syntax box indicate that you can reset more than one mode in a single RM command by stringing parameters together, separated by semicolons.

The RM command resets the modes you set with the SM (SET MODE) command. See the SM command description for details about these modes and how to issue more than one parameter at a time.

NOTE

When the terminal encounters a parameter beginning with a prefix (? or >), it uses the same prefix for all subsequent digit-only parameters.

This means that if you issue an RM command with more than one parameter, you should issue the digit-only parameters first, followed by any prefixed parameters.

When you reset more than one mode and the first parameter you specify begins with a prefix (< or ?), the terminal interprets all subsequent digit-only parameters as also beginning with that prefix.

If you are issuing a series of parameters that all start with the ? prefix, you can issue the first parameter only with the ?, and omit the ? from subsequent parameters.

For compatibility with other manufacturer's terminals, you should use one RM (RESET MODE) command to set any modes with prefixed parameters and another RM command to reset any modes whose parameters consist of digits only.

For example, do not mix digit-only parameters and prefixed parameters like this:

```
Ec[?3;4;5I
```

Issue two commands instead:

```
Ec[?3;5I
Ec[4I
```

Syntax Example

```
Host: Ec[4;20I
Setup: INSERTREPLACE REPLACE
      LFCR YES
```

Sets Insert/Replace mode to Replace and specifies that a Line Feed (LF) implies a Line Feed/Carriage Return combination (LFCR).

SCS (SELECT CHARACTER SET)

SCS (SELECT CHARACTER SET)

Selects one or two character sets from the eight stored in the terminal's firmware and makes them available through the keyboard.

Host Syntax (to select G0)

```
Esc character-set
```

Host Syntax (to select G1)

```
Esc character-set
```

Setup Syntax (to select G0)

```
SELECTCHARSET G0 character-set
```

Setup Syntax (to select G1)

```
SELECTCHARSET G1 character-set
```

character-set: specifies the character set you want. Valid values are shown in Table 3-5.

Defaults: Factory = Determined by keyboard
 Omitted = (none)

The terminal allows you to access two different character sets by using the SI (SHIFT IN) and SO (SHIFT OUT) commands to select either the currently defined G0 or G1 character set.

When the terminal is turned on, the character set associated with the particular keyboard is designated as the both the G0 character set and the G1 character set. This command allows you to designate different G0 or G1 sets.

This command controls the character set used to display data transmitted from the host or typed in Local mode; it doesn't affect the characters displayed in Setup. If you are using Local mode, you must enable local echo to display the characters you type.

Table 3-5 shows the parameters needed to select a particular character set. Appendix A contains tables that list the contents of each character set.

Table 3-5

SCS (SELECT CHARACTER SET) VALUES

Value	Character Set Designated
A	United Kingdom
B	ASCII/North American
G	Swedish
K	German
f (or R)	French (see note)
'	Danish/Norwegian
0	Rulings Set
3	Supplementary

NOTE

While the terminal will accept R as a parameter value to select the French character set, the current standard is f. So for compatibility with current and future standards, you should use f to select the French character set.

Syntax Example

Host: EscA
 Setup: SELECTCHARSET G1,A

Selects the United Kingdom character set as the G1 character set.

SD (SCROLL DOWN)

Scrolls lines down.

Host Syntax

```
Ec[ number-of-lines T
```

number-of-lines: specifies the number of lines the dialog buffer scrolls toward the bottom of the screen.

Defaults: Factory = (none)
 Omitted or 0 = 1

The SD command shifts all lines displayed on the screen down the specified number (*n*) of rows. The *n* lines at the bottom margin are rolled out of sight and *n* lines are rolled into view at the top margin.

Syntax Example

```
Ec[8T
```

Scrolls down eight lines.

SELECT CODE

Selects the host command mode, choosing ANSI, EDIT, VT52, or TEK host command syntax. (Can be saved in nonvolatile memory.)

Host Syntax

```
Ec%!syntax
```

Setup Syntax

```
CODE syntax
```

syntax: specifies the host command mode that you want to use:

Host	Setup	
0	TEK	Selects TEK mode
1	ANSI	Selects ANSI mode
2	EDIT	Selects EDIT mode
3	VT52	Selects VT52 mode

Defaults: Factory = (none)
 Omitted = TEK

This command causes the terminal to accept ANSI, EDIT, TEK, or VT52 commands from the host computer. The syntax of these different host command modes are not compatible. If you are using host commands from one mode and want to execute one or more commands from another mode, you must issue the SELECT CODE command with the appropriate parameter. This command is recognized in all host command modes.

Although this command does not affect the availability of commands entered from the keyboard in Setup, you can issue it while in Setup to cause the terminal to recognize a specific command syntax when it leaves Setup.

See the discussions *ANSI mode*, *EDIT mode*, and *VT52 mode* earlier in this Section for more information on the effect of this command on screen editing programs.

Selecting EDIT mode resets a number of terminal characteristics to emulate a VT100 terminal. When you select TEK mode after using EDIT mode, these terminal settings may not be appropriate, and you may need to reset them so your program will work as expected. See the command description for SELECT CODE in Section 5 for details.

Syntax Example

```
Host: Ec%!2  

Setup: CODE EDIT
```

Selects EDIT mode.

SGR (SELECT GRAPHIC RENDITION)

Selects display attributes for text in the dialog area.

Host Syntax

```
Esc[ graphic-rendition . . . m
```

Setup Syntax

```
TEXTRENDITION graphic-rendition . . .
```

graphic-rendition: specifies the colors and other display characteristics for text displayed in the dialog area.

Defaults: Factory = 0
 Omitted = 0

The SGR command controls colors and other display characteristics in the dialog area.

The colors you can control with SGR are:

- *Character color* — the color of the characters as they are displayed on the screen
- *Character background color* — the color of the character cell which surrounds each character
- *Dialog Area Background color* — the color of the dialog area background before anything is written on it

You can emphasize characters by selecting blinking display, bold display, underscored display, or reverse video.

When you select a color or other display characteristic, previously entered text will be unaffected, and only newly entered text will have the new color or characteristic.

You can control character color, character background color, and background color by using the 4100-style command SET DIALOG AREA INDEX.

Table 3-6

DIGIT-ONLY PARAMETERS FOR THE SGR COMMAND

Display Characteristic	Parameter	Action
All color indices	0	Returns color indices to values set by SET DIALOG AREA INDEX command
Character emphasis	1	Simulates bold characters by displaying text in Index 2 (default <i>red</i>)
	4	Starts underscoring text
	5	Starts blinking text
	7	Reverses character and character background indices
	24	Stops underscoring characters
	25	Stops blinking characters
Character color	27	Returns character and character background indices to original values
	30	Selects Index 0 (default <i>black</i>)
	31	Selects Index 2 (default <i>red</i>)
	32	Selects Index 3 (default <i>green</i>)
	33	Selects Index 7 (default <i>yellow</i>)
	34	Selects Index 4 (default <i>blue</i>)
	35	Selects Index 6 (default <i>magenta</i>)
	36	Selects Index 5 (default <i>cyan</i>)
	37	Selects Index 1 (default <i>white</i>)
	39	Selects Index 1 (default <i>white</i>)
Character background color	40	Selects Index 0 (default <i>black</i>)
	41	Selects Index 2 (default <i>red</i>)
	42	Selects Index 3 (default <i>green</i>)
	43	Selects Index 7 (default <i>yellow</i>)
	44	Selects Index 4 (default <i>blue</i>)
	45	Selects Index 6 (default <i>magenta</i>)
	46	Selects Index 5 (default <i>cyan</i>)
	47	Selects Index 1 (default <i>white</i>)
	49	Selects Index 0 (default <i>transparent</i>)

About SGR Parameters. There are two groups of parameter values available for the SGR command:

- *Digit-only parameters.* The parameter values 0 through 49 control character color, character background color, and other characteristics that emphasize text, such as blinking and underscoring. These parameters *cannot* be saved in nonvolatile memory. The display characteristics selected by these parameters are listed in Table 3-6.
- *Prefixed parameters.* These parameter values consist of a prefix (<, >, or =) and an index. The prefix controls where the color will be used, and the index is an integer that specifies which color will be used. These parameters are not available in Setup. They *can* be saved in nonvolatile memory. The display characteristics selected by these parameters are summarized in Table 3-7.

Table 3-7

PREFIXED PARAMETERS FOR THE SGR COMMAND

Display Characteristic	Parameter ^{a,b}	Action
Character color	< <i>index</i>	Specifies the character index. Index 0 selects black characters.
Character background color	= <i>index</i>	Specifies the character background index. Index 0 means that the graphics area shows through.
Dialog area background color	> <i>index</i>	Specifies the background index. Index 0 means that the graphics area shows through.

^a These parameters are available in host syntax only; they cannot be issued in Setup.

^b *index* is a variable — you fill in a number from 0 to 7 to specify a color.

As indicated by the three dots (. . .) in the syntax boxes, you can set more than one display attribute by entering more than one parameter value.

NOTE

When the terminal encounters a parameter beginning with a prefix (<, =, or >), it uses the same prefix for all subsequent digit-only parameters.

This means that if you issue an SGR command with more than one parameter, you should issue the digit-only parameters first, followed by any prefixed parameters.

In host syntax, you should use one SGR (Select Graphic Rendition) command to set the < *index*, = *index*, and > *index* parameter values and another SGR command to reset any modes whose parameter values consist of digits only. You can avoid the complications of mixing the two types of parameters in host syntax by using only parameters 0 through 49, which can set all the available graphic rendition features, except background color, and can be mixed freely.

Either of these methods will make your program compatible with most ANSI terminals.

Syntax Example

In host syntax you could select red underlined characters by issuing:

`Ec[<2m` (Selects Index 2, red)
`Ec[4m` (Selects underlining)

Or, more simply, by issuing:

`Ec[4;31m` (4 selects underlining; 31 selects red display)

In Setup syntax, you could select red underlining by issuing:

TEXTRENDITION 4,31

(You don't have to worry about mixing parameter types in Setup syntax, since only parameters 0 through 49 are valid.)

SI (SHIFT IN)

SI (SHIFT IN)

Invokes the current G0 character set.

Host Syntax

```
SI
```

The terminal allows you to access two different character sets by using the SI (SHIFT OUT) and SO (SHIFT OUT) commands to switch between the currently defined G0 or G1 character sets.

The SI command invokes the currently defined G0 character set. This may be the 94 graphic characters from the ASCII character set, or the corresponding 94 characters from the United Kingdom, French, Swedish, Danish/Norwegian, German, supplementary, or special rulings character sets. Changing the keyboard selects the corresponding character set automatically, but all character sets are available. You can use the SCS (SELECT CHARACTER SET) command to choose the character set you want no matter which keyboard is connected. Appendix A of this manual lists these character sets.

To select the G1 character set, use the SO (SHIFT OUT) command.

SL (SCROLL LEFT)

Moves the visible columns of the dialog area to the left.

Host Syntax

```
Ec[ number-of-columnsSP@
```

number-of-columns: specifies the number of columns the dialog buffer scrolls to the left.

Defaults: Factory = (none)
Omitted or 0 = 1

The SL command moves the entire contents of the visible portion of the dialog area to the left by the specified number of columns. You can scroll horizontally only when Column mode is set to 132. Since the cursor moves with the text, the cursor may disappear from the screen.

The terminal will not automatically scroll left or right to keep the cursor in view. To scroll horizontally, you must give the SL (SCROLL LEFT) or SR (SCROLL RIGHT) command (or you can use the Joydisk).

When you make the dialog area visible, the terminal uses scrolling to bring the cursor into view. If the cursor is in Columns 1 through 80, the dialog area will scroll right to bring the leftmost column into view; if the cursor is in Columns 81 through 132, the dialog area will scroll left to bring the rightmost column into view.

Syntax Example

```
Ec[12SP@
```

Scrolls 12 columns to the left.

SM (SET MODE)

Sets one or more terminal modes — used with the RM (RESET MODE) command.

Host Syntax

```
Ec[ mode . . . h
```

Setup Syntax

(See Table 3-8)

mode: sets one or more of the following ANSI modes. Most of these modes can be invoked by their own Setup command, shown in Table 3-8 (next page); you can also look each mode up separately under its mode name. Valid values are:

- 2 KAM (Keyboard Action Mode)
- 4 IRM (Insert/Replace Mode)
- 12 SRM (Send/Receive Mode)
- 20 LNM (Linefeed/Newline Mode)
- <1 TEKORM (Overstrike/Replace Mode)
- ?1 TEKCKM (Cursor Keys Mode)
- ?3 TEKCOLM (Column Mode)
- ?5 TEKSCNM (Screen Mode)
- ?6 TEKOM (Origin Mode)
- ?7 TEKAWM (Autowrap Mode)
- ?8 TEKARM (Autorepeat Mode)

Defaults: Factory = (none)
 Omitted = Error [h11

The SM command sets one or more modes; each mode remains set until you reset it with an RM (RESET MODE) command.

The three periods (. . .) in the host syntax box indicate that you can reset more than one mode in a single RM command by stringing parameters together, separated by semicolons.

NOTE

When the terminal encounters a parameter beginning with a prefix (? or <), it uses the same prefix for all subsequent digit-only parameters.

This means that if you issue an SM command with more than one parameter, you should issue the digit-only parameters first, followed by any prefixed parameters.

When you set more than one mode and the first parameter you specify begins with a prefix (< or ?), the terminal interprets all subsequent digit-only parameters as also beginning with that prefix.

If you are issuing a series of parameters that all start with the ? prefix, you can issue the first parameter only with the ?, and omit the ? from subsequent parameters.

For compatibility with other manufacturer's terminals, you should use one SM (SET MODE) command to set any modes with prefixed parameters and another SM command to reset any modes whose parameters consist of digits only.

For example, do not mix digit-only parameters and prefixed parameters like this:

```
Ec[4;?3;5h
```

Issue two commands instead:

```
Ec[?3;5h
Ec[4h
```

(continued)

SM (SET MODE)

The following paragraphs describe the modes you can control with RM and SM; Table 3-8 summarizes this information and gives the host and Setup syntax to set and reset each terminal mode.

Keyboard Action Mode (KAM). SM disables the terminal keyboard. RM returns the terminal to its default state by enabling the terminal keyboard.

Insert/Replace Mode (IRM). SM causes each entered character to be inserted at the cursor position and characters to the right of the cursor position move to the right. RM returns the terminal to its default state, in which each character entered overwrites (that is, *replaces*) the character at the cursor position.

Send/Receive Mode (SRM). SM enables local echo — that is, the terminal displays characters as they are sent to the host. RM returns the terminal to its default state by disabling local echo (this is appropriate if the host provides an echo).

Linefeed/Newline Mode (LNM). SM causes a LF character to also imply a CR (Carriage Return) character, and so moves the cursor to the beginning of the next line. RM returns the terminal to its default state, in which a LF (Line Feed) character moves the cursor down one line without changing its column position.

Overstrike/Replace Mode (TEKORM). SM causes the Underscore character to underline the current character and the Space character to move the cursor forward one column (erasing the underscore if one is present). RM returns the terminal to its default state, in which the Space character and the Underscore character overwrite characters at the current position.¹

Cursor Keys Mode (TEKCKM). SM causes Function Keys F1 through F4 to transmit application program codes. RM returns the terminal to its normal state, in which Function Keys F1 through F4 transmit regular ANSI cursor-control commands (in the terminal's normal state, if these keys are programmed and key expansions are enabled, they transmit their programmed values). Table 3-9 shows the codes that these keys transmit when reset and set.

ANSI-to-VT52 Mode (TEKANM). SM has no effect on ANSI-to-VT52 mode. RM puts the terminal in VT52 mode.

¹ Unless Insert/Replace mode (IRM) is set to *insert*.

Column Mode (TEKCOLM). SM specifies 132-column width. RM returns the terminal to its default 80-column width. Setting or resetting this mode erases the contents of the dialog area and resets the edit margins to the top and bottom lines of the dialog area; setting and resetting this mode does not affect Origin mode, tabs, character attributes, or any other screen attributes.

When Column mode is set to 132, the terminal displays only 80 of the 132 columns at any time. You can use the SL (SCROLL LEFT) or SR (SCROLL RIGHT) commands or the Joydisk to scroll horizontally to bring any of the columns into view.

When the cursor is off the screen to the left or right, you can use scrolling commands to bring it back; making the dialog area visible (with the Dialog key or SET DIALOG AREA VISIBILITY command) also scrolls the dialog buffer to bring the cursor into view. If the cursor is in Columns 1 through 80, the dialog buffer will scroll right to bring the leftmost column into view; if the cursor is in Columns 81 through 132, the dialog buffer will scroll left to bring the rightmost column into view.

Screen Mode (TEKSCNM). SM reverses the colors on the display (in terms of the HLS color coordinate system, adds 180 to the first color coordinate) and makes Index 0 in the dialog area opaque. RM returns the terminal to its default state, setting the colors on the display to their normal values and making Index 0 in the dialog area transparent.

Origin Mode (TEKOM). SM sets Origin mode to Absolute (the cursor moves to Row 1, Column 1 of the dialog buffer; cursor addressing is relative to Row 1, Column 1 of the dialog buffer; and the dialog buffer size is reduced to the screen size). RM returns the terminal to its default, Origin mode Relative (if edit margins have been set, the cursor will move to Row 1, Column 1 of the scrolling region instead of to the corresponding position in the dialog buffer).

Autowrap Mode (TEKAWM). SM enables autowrap so that characters entered in the rightmost column wrap around to the next line (this is the default). RM disables autowrap so that characters entered in the rightmost column write over existing characters in that column.

Autorepeat Mode (TEKARM). SM enables autorepeat so that keyboard keys repeat when held down (this is the default). RM disables autorepeat so that keyboard keys do not repeat when held down.

Syntax Example

```
Host:  Ec[4;20h
Setup: INSERTREPLACE insert
      LFCR yes
```

Sets Insert/Replace mode to Insert and specifies that a Line Feed LF also implies a Carriage Return CR .

Table 3-8
RM (RESET MODE) AND SM (SET MODE) COMMAND PARAMETERS

Mode Name ^a	Action	Host Syntax	Setup Syntax
IRM (INSERT/REPLACE MODE)	<i>Reset:</i> Replace	E _c [4l	INSERTREPLACE replace
	<i>Set:</i> Insert	E _c [4h	INSERTREPLACE insert
KAM (KEYBOARD ACTION MODE)	<i>Reset:</i> Enables keyboard	E _c [2l	(none)
	<i>Set:</i> Disables keyboard	E _c [2h	(none)
LNM (LINEFEED/NEWLIN MODE)	<i>Reset:</i> Line Feed only	E _c [20l	LFCR no
	<i>Set:</i> Line Feed and Carriage Return	E _c [20h	LFCR yes
SRM (SEND/RECEIVE MODE)	<i>Reset:</i> Enables echo	E _c [12l	ECHO yes
	<i>Set:</i> Disables echo	E _c [12h	ECHO no
TEKANM (ANSI-TO-VT52 MODE)	<i>Reset:</i> Selects VT52 mode	E _c [?2l	CODE VT52
	<i>Set:</i> No effect	(none)	(none)
TEKARM (AUTOREPEAT MODE)	<i>Reset:</i> Disables autorepeat	E _c [?8l	AUTOREPEAT no
	<i>Set:</i> Enables autorepeat	E _c [?8h	AUTOREPEAT yes
TEKAWM (AUTOWRAP MODE)	<i>Reset:</i> Disables autowrap	E _c [?7l	AUTOWRAP no
	<i>Set:</i> Enables autowrap	E _c [?7h	AUTOWRAP yes
TEKCKM (CURSOR KEYS MODE)	<i>Reset:</i> Function Keys F1 — F4 transmit normal commands or programmed values	E _c [?1l	CURSORKEYMODE no
	<i>Set:</i> Function Keys F1 — F4 transmit application values	E _c [?1h	CURSORKEYMODE yes
TEKCOLM (COLUMN MODE)	<i>Reset:</i> Specifies 80 column dialog buffer	E _c [?3l	COLUMNMODE 80
	<i>Set:</i> Specifies 132 column dialog buffer	E _c [?3h	COLUMNMODE 132
TEKOM (ORIGIN MODE)	<i>Reset:</i> Cursor address Row 1, Column 1 is beginning of dialog buffer	E _c [?6l	ORIGINMODE absolute
	<i>Set:</i> Cursor address Row 1, Column 1 is beginning of scrolling region	E _c [?6h	ORIGINMODE relative
TEKORM (OVERSTRIKE/REPLACE MODE)	<i>Reset:</i> Space and Underscore replace existing characters	E _c [<1l	DAMODE replace
	<i>Set:</i> Underscore underlines existing characters and Space moves the cursor forward one space	E _c [<1h	DAMODE overstrike
TEKSCNM (SCREEN MODE)	<i>Reset:</i> Normal colors; Index 0 transparent	E _c [?5l	SCREENMODE normal
	<i>Set:</i> Reverse colors; Index 0 opaque	E _c [?5h	SCREENMODE reverse

^a You can also look up each of these modes under its mode name alphabetically in these command descriptions.

Table 3-9
CURSOR KEY MODE CODES

Function Key	Codes Sent When Set (SM)	Codes Sent When Reset (RM)
F1	E _c OA	E _c [A
F2	E _c OB	E _c [B
F3	E _c OD	E _c [D
F4	E _c OC	E _c [C

SO (SHIFT OUT)

Invokes the G1 character set.

Host Syntax

```
SO
```

The terminal allows you to access two different character sets by using the SI (SHIFT IN) and SO (SHIFT OUT) commands to switch between the currently defined G0 and G1 character sets.

The SO command invokes the G1 character set. When a keyboard is plugged into the terminal, the character set associated with that keyboard is designated as both the G0 and the G1 set. You may use the SCS (SELECT CHARACTER SET) command to designate a different character set than the one associated with the current keyboard. Appendix A lists the contents of all the available character sets.

To select the G0 character set, use the SI (SHIFT IN) command.

SR (SCROLL RIGHT)

Moves the visible columns of the dialog area to the right.

Host Syntax

```
EC[ number-of-columns SP A
```

number-of-columns: specifies the number of columns the dialog buffer scrolls to the right.

Defaults: Factory = (none)
Omitted or 0 = 1

The SR command moves the entire contents of the visible portion of the dialog area to the right by the specified number of columns. Since the cursor moves with the text, the cursor may disappear from the screen. Unlike vertical scrolling, the terminal will not automatically scroll left or right to keep the cursor in view. To scroll horizontally, you must give the SL (SCROLL LEFT) or SR (SCROLL RIGHT) command (or you can use the Joydisk).

When you make the dialog area visible, the terminal uses scrolling to bring the cursor into view. If the cursor is in Columns 1 through 80, the dialog area will scroll right to bring the leftmost column into view; if the cursor is in Columns 81 through 132, the dialog area will scroll left to bring the rightmost column into view.

Syntax Example

```
EC[12SPA
```

Scrolls right 12 columns.

SRM (SEND/RECEIVE MODE)

Specifies whether the terminal provides its own echo (local echo) of data entered at the keyboard. (Can be saved in nonvolatile memory.)

Setup Syntax

```
ECHO mode
```

mode: keyword; specifies whether the terminal provides its own echo. Valid values are: *no* and *yes*.

Defaults: Factory = no
Omitted = yes

This command has the same effect as the 4100-style SET ECHO command.

If Send/Receive mode is selected, the terminal displays characters as they are sent to the host (this is local echo). Otherwise, the terminal does not echo locally.

The SRM command is part of the RM (RESET MODE) and SM (SET MODE) commands. See the description of the RM and SM commands to see how to change the settings of the SRM command from the host.

SU (SCROLL UP)

Scrolls lines up.

Host Syntax

```
Ec[ number-of-lines S
```

number-of-lines: specifies the number of lines the dialog buffer scrolls toward the top of the screen.

Defaults: Factory = (none)
Omitted or 0 = 1

The SU (SCROLL UP) command shifts all lines displayed on the screen upward by the specified number (*n*) of rows. The *n* lines at the top margin are rolled out of sight and *n* lines are rolled into view at the bottom margin.

Syntax Example

```
Ec[12S
```

Scrolls up 12 lines.

SUB (SUBSTITUTE)

Cancels an ANSI command in progress.

Host Syntax

```
SB
```

When the terminal receives this character, it cancels any ANSI command currently being processed and inserts a S_B character at the current cursor location in the dialog area.

TBC (TAB CLEAR)

Clears one or more tab stops.

Host Syntax

```
Ec[ tab-clear-extent g
```

tab-clear-extent: specifies how many tab stops to clear:

- 0 Clears the horizontal tab stop at the cursor position
- 2 Clears all horizontal tab stops
- 3 Clears all horizontal tab stops

Defaults: Factory = (none)
Omitted = 0

Syntax Example

```
Ec[2g
```

Clears all horizontal tab stops.

TEKANM (ANSI-TO-VT52 MODE)

Selects VT52 mode.

Setup Syntax

```
CODE VT52
```

This command switches the terminal's host command mode from ANSI to VT52 mode — it has the same effect as the 4100-style SELECT CODE command issued with the VT52 parameter.

The TEKANM command is part of the RM (RESET MODE) and SM (SET MODE) commands. See the description of the RM and SM commands to see how to change the settings of the TEKANM command from the host.

TEKARM (AUTOREPEAT MODE)

Specifies whether keys on the terminal's keyboard repeat when held down. (Can be saved in nonvolatile memory.)

Setup Syntax

```
AUTOREPEAT mode
```

mode: keyword; specifies whether terminal keys repeat when held down. Valid values are: *no* and *yes*.

Defaults: Factory = yes
Omitted = yes

When Autorepeat mode is selected, keyboard keys repeat when held down. Otherwise, keyboard keys do not repeat when held down.

The TEKARM command is part of the RM (RESET MODE) and SM (SET MODE) commands. See the description of the RM and SM commands to see how to change the settings of the TEKARM command from the host.

TEKAWM (AUTOWRAP MODE)

Specifies whether characters written to the rightmost column overwrite existing characters or wrap to the next line. (Can be saved in nonvolatile memory.)

Setup Syntax

```
AUTOWRAP mode
```

mode: keyword; specifies whether or not characters wrap to next line. Valid values are: *no* and *yes*.

Defaults: Factory = yes
Omitted = yes

When Autowrap mode is selected, characters entered in the rightmost column wrap around to the next line. Otherwise, characters entered in the rightmost column write over existing characters in that column.

The TEKAWM command is part of the RM (RESET MODE) and SM (SET MODE) commands. See the description of the RM and SM commands to see how to change the settings of the TEKAWM command from the host.

TEKCKM (CURSOR KEYS MODE)

Specifies whether or not the F1 through F4 function keys transmit ANSI cursor control commands.

Setup Syntax

CURSORKKEYMODE mode

mode: keyword; specifies whether the F1 through F4 function keys transmit cursor control commands. Valid values are: *no* and *yes*.

Defaults: Factory = no
Omitted = yes

When Cursor Keys mode is selected (*CURSORKKEYMODE YES*), Function Keys F1 through F4 transmit regular ANSI cursor-control commands; however, if these keys are programmed and key expansions are enabled, they transmit their programmed values. Otherwise, Function Keys F1 through F4 transmit application program codes. Table 3-10 shows the codes that these keys transmit in either case.

The TEKCKM command is part of the RM (RESET MODE) and SM (SET MODE) commands. See the description of the RM and SM commands to see how to change the settings of the TEKCKM command from the host.

Table 3-10
CURSOR KEYS MODE CODES

Function Key	Codes Sent When Cursor Keys Mode is Set to Yes	Codes Sent When Cursor Keys Mode is Set to No
F1	E_cOA	$E_c[A$
F2	E_cOB	$E_c[B$
F3	E_cOD	$E_c[D$
F4	E_cOC	$E_c[C$

TEKCOLM (COLUMN MODE)

Selects 80- or 132-column width for the dialog buffer. (Can be saved in nonvolatile memory.)

Setup Syntax

COLUMNMODE mode

mode: keyword; specifies the width of the dialog buffer. Valid values are: *80* and *132*.

Defaults: Factory = 80
Omitted = 80

Setting and resetting this mode erases the contents of the dialog area and resets the edit margins to the top and bottom lines of the dialog area; setting and resetting this mode does not affect Origin mode, tabs, character attributes, or any other screen attributes.

When Column mode is set to 132, the terminal displays only 80 of the 132 columns at any time. When the cursor is off the screen to the left or right, you can use scrolling commands to bring it back; making the dialog area visible (with the Dialog key or SET DIALOG AREA VISIBILITY command) also scrolls the dialog buffer to bring the cursor into view. If the cursor is in Columns 1 through 80, the dialog buffer will scroll right to bring the leftmost column into view; if the cursor is in Columns 81 through 132, the dialog buffer will scroll left to bring the rightmost column into view.

The TEKCOLM command is part of the RM (RESET MODE) and SM (SET MODE) commands. See the description of the RM and SM commands to see how to change the settings of the TEKCOLM command from the host.

TEKDHL (DOUBLE HEIGHT LINE)

Causes the line containing the cursor to become the top or bottom half of a double-height, double-width line.

Host Syntax

<u>Top Half</u>	<u>Bottom Half</u>
E _C #3	E _C #4

Both lines that receive these commands must contain the same characters. Since using double-width characters halves the number of characters per line, characters to the right of screen center are lost if the line was previously single width.

If the terminal receives the Bottom Half command without receiving the Top Half command first, the line will be double-width and single-height.

To make an exact hardcopy of a double-height, double-width line, you must make a screen copy (use the **HARDCOPY** command with a parameter of 0 or use the **S Copy** key). Making a dialog copy (use the **HARDCOPY** command with a parameter of 3 or use the **D Copy** key) will copy each character of the top-half line as a regular size character followed by a space; the bottom-half line becomes a blank line. (See the command description for the 4100-style **HARDCOPY** command for additional details about making screen and dialog copies.)

This command affects only the current line. The line will retain this attribute until the line is deleted or until the terminal receives another line attribute command (TEKDHL, TEKDWL, or TEKSWL).

TEKDWL (DOUBLE WIDTH LINE)

Causes the line containing the cursor to become a double-width, single-height line.

Host Syntax

E _C #6

This command affects only the current line. The line will retain this attribute until the line is deleted or until the terminal receives another line attribute command (TEKDHL, TEKDWL, or TEKSWL).

Since using double-width characters halves the number of characters available per line, characters to the right of screen center are lost if the line was previously single width.

To make an exact copy of a double-width line, you must make a screen copy. Making a dialog copy will copy each character in the line as a regular size character followed by a space. (See the command description for the 4100-style **HARDCOPY** command for additional details about making screen and dialog copies.)

TEKID (IDENTIFY TERMINAL)

Tells the terminal to report what type of terminal it is.

Host Syntax

E _C Z

Report Syntax

E _C [?1;2c

This command causes the same response as the ANSI command **DA** (DEVICE ATTRIBUTES) with a parameter of 0.

NOTE

The TEKID command is provided in ANSI mode only for compatibility with programs written for VT100 terminals. Avoid using this command if you can; its use violates ANSI and ISO standards.

TEKKPAM (KEYPAD APPLICATION MODE)

Causes the numeric keypad and Function Keys F5 through F8 to send special escape sequences.

Host Syntax

$E_C =$

Setup Syntax

KEYPADMODE APPLICATION

The TEKKPAM command causes the numeric keypad to send characters distinct from the numeric keys on the main keyboard. This means that when you press the 6 key on the numeric keypad, a different code is generated than when you press the 6 key on the main keyboard. Refer to Table 3-11 for an explanation of these codes.

When the terminal is turned on, it is in Keypad Numeric mode.

TEKKPNM (KEYPAD NUMERIC MODE)

Causes the numeric keypad and Function Keys F5 through F8 to send their default values.

Host Syntax

$E_C >$

Setup Syntax

KEYPADMODE NUMERIC

This command causes the keys on the numeric keypad and Function Keys F5 through F8 to return to their default meanings, as shown in the righthand column of Table 3-11. If the keys have been programmed and key expansions are enabled, the keys transmit their programmed meanings instead.

When the terminal is turned on, it is in Keypad Numeric mode (keys produce their default meanings).

Table 3-11
NUMERIC KEYPAD PROGRAMMING CODES^a

Numeric Keypad Key	Characters Sent in Application Mode	Characters Sent in Numeric Mode ^b (Default)
0	E_cOp	0
1	E_cOq	1
2	E_cOr	2
3	E_cOs	3
4	E_cOt	4
5	E_cOu	5
6	E_cOv	6
7	E_cOw	7
8	E_cOx	8
9	E_cOy	9
-	E_cOm	-
,	E_cOl	,
.	E_cOn	.
ENTER	E_cOM	C_R
F5	E_cOP	E_cOP
F6	E_cOQ	E_cOQ
F7	E_cOR	E_cOR
F8	E_cOS	E_cOS

^a Refer to the discussion *VT52 Mode*, earlier in this section for an explanation of how VT52 mode commands affect codes sent by the keypad and function keys.

^b If these keys are programmed with macros and you haven't disabled key expansion, the terminal sends the macros rather than the characters listed in this column.

TEKOM (ORIGIN MODE)

Specifies how the terminal interprets cursor addresses in ANSI commands. (Can be saved in nonvolatile memory.)

Setup Syntax

```
ORIGINMODE mode
```

mode: keyword; specifies the way the terminal interprets cursor addresses. Valid values are: *absolute* and *relative*.

Defaults: Factory = relative
Omitted = relative

When Origin mode Absolute is selected; the cursor moves to Row 1, Column 1 of the dialog buffer; cursor addressing is based on Row 1, Column 1 of the dialog buffer; and the dialog buffer size is reduced to the screen size.

When Origin mode Relative is selected and edit margins are set, the cursor will move to Row 1, Column 1 of the scrolling region instead of to the corresponding position in the dialog buffer.

The TEKOM command is part of the RM (RESET MODE) and SM (SET MODE) commands. See the description of the RM and SM commands to see how to change the settings of the TEKOM command from the host.

TEKORM (OVERSTRIKE/REPLACE MODE)

Controls how the terminal displays Underscore and Space characters sent to the terminal screen. (Can be saved in nonvolatile memory.)

Setup Syntax

```
DAMODE mode
```

mode: keyword; specifies the way the terminal treats the Space (␣) and Underscore (␣) characters. Valid values are: *overstrike* and *replace*.

Defaults: Factory = replace
Omitted = replace

Use this command with screen editing programs that rely on a printer's overstrike capability to create underscoring. TEKORM allows you to display underscoring in formatted files so that it looks the same on the screen as it does on a hard copy. Screen editing programs that turn underscoring on and off with the ANSI command SGR (SELECT GRAPHIC RENDITION) do not need to use the TEKORM command to emulate underscoring on the terminal screen.

When Overstrike/Replace mode is set to *replace* (which is the terminal's factory default), the Space and Underscore characters overwrite other characters¹, as they normally do. When Overstrike/Replace mode is set to *overstrike*, the terminal treats Space and Underscore in the same way as a printer does — the Underscore character underlines the current character and the Space character just moves the cursor forward without erasing characters. (On the screen, however, the Space character erases underscores.)

The TEKORM command is part of the RM (RESET MODE) and SM (SET MODE) commands. See the description of the RM and SM commands to see how to change the settings of the TEKORM command from the host. The 4100-style command SET DIALOG AREA WRITING MODE also controls the Space and Underscore characters in the same way as the TEKORM command.

¹ Unless Insert/Replace mode (IRM) is set to *insert*.

TEKRC (RESTORE CURSOR)

Restores the cursor position, graphic rendition, character set, and Origin mode previously saved using the TEKSC (SAVE CURSOR) command.

Host Syntax

```
Ec8
```

If the TEKSC (SAVE CURSOR) command is not used first, TEKRC (RESTORE CURSOR) returns the cursor to the Home position (Row 1, Column 1 of the dialog buffer) and restores the power-up graphic rendition, character set, and Origin mode.

TEKSC (SAVE CURSOR)

Saves the cursor position, graphic rendition, character set, and Origin mode.

Host Syntax

```
Ec7
```

The TEKSC (SAVE CURSOR) command temporarily saves information about the cursor position, graphic rendition, character set, and Origin mode in the terminal's program memory. This saved information may be restored using the TEKRC (RESTORE CURSOR) command.

TEKSCNM (SCREEN MODE)

Displays colors in the dialog area in normal or reversed values. (Can be saved in nonvolatile memory.)

Setup Syntax

```
SCREENMODE mode
```

mode: keyword; specifies the way the terminal displays color indices in the dialog area. Valid values are: *normal* and *reverse*.

Defaults: Factory = normal
Omitted = normal

When Screen mode is *normal*, colors in the dialog area have their normal values and Index 0 in the dialog area is transparent. When Screen mode is *reverse*, the terminal reverses the colors in the dialog area (in terms of the HLS color coordinate system, it adds 180 to the first color coordinate) and makes Index 0 in the dialog area opaque.

The SGR (SELECT GRAPHICS RENDITION) ANSI command can also be used to reverse colors. The SGR command reverses the colors in the graphics area and in the dialog area.

The TEKSCNM command is part of the RM (RESET MODE) and SM (SET MODE) commands. See the description of the RM and SM commands to see how to change the settings of the TEKSCNM command from the host.

TEKSTBM (SET TOP AND BOTTOM MARGINS)

Sets the dialog buffer's edit margins.

Host Syntax

```
Ec[ top-margin ; bottom-margin r
```

Setup Syntax

```
EDITMARGIN top-margin,bottom-margin
```

top-margin: specifies the top margin of the scrolling region.

Defaults: Factory = (none)
Omitted or 0 = 1

bottom-margin: specifies the the bottom margin of the scrolling region.

Defaults: Factory = (none)
Omitted or 0 = last line of dialog area

The value for the top margin specifies which row of the dialog buffer becomes the top line of the scrolling region. Similarly, the value for the bottom margin specifies the row of the dialog buffer for the bottom line of the scrolling region.

The rows in the dialog buffer above the top margin and the rows below the bottom margin become fixed regions. No scrolling actions occur in the fixed regions.

If the dialog buffer is greater than the screen size and you set edit margins other than the screen margins (Row 1 and Row 30), issuing the TEKSTBM command will reduce the number of lines in the dialog buffer to match the screen size.

Syntax Example

Host: E_c[5;15r
Setup: EDITMARGINS 5,15

Sets the edit margins at Rows 5 and 15.

TEKSWL (SINGLE WIDTH LINE)

Causes the current line to become a single-width, single-height line.

Host Syntax

```
Ec#5
```

The cursor retains its current column number. This is the default for all new lines in the dialog area. This command affects only the current line. The line will retain this attribute until the line is deleted or until the terminal receives another line attribute command (TEKDHL, TEKDWL, or TEKSWL).

VT (VERTICAL TAB)

Moves the cursor down one line without affecting the cursor position on the line.

Host Syntax

```
vT
```


VT52 COMMAND DESCRIPTIONS

This part of Section 3 contains descriptions of the terminal's VT52 commands. The commands are presented alphabetically according to their descriptive names.

The VT52 commands that follow can be executed only while the terminal is in VT52 mode. You can put the terminal in VT52 mode by:

- Entering **CODE VT52** while in Setup
- Sending an RM command ($\text{Ec}[?2\text{I}]$) from the host while in ANSI mode
- Sending a SELECT CODE command ($\text{Ec}\%!\text{3}$) from the host while in TEK or ANSI mode

Once the terminal is in VT52 mode, it will recognize only VT52 commands (which are explained here), and the commands SELECT CODE, ENQUIRY, and REPORT SYNTAX MODE, all of which work in all host command modes.

CURSOR DOWN

Moves the cursor down one line without moving it horizontally.

Host Syntax

EcB

The cursor address is based on the first line of the dialog buffer (Row 1, Column 1 is the first position in the buffer), and the cursor stops at the last row of the dialog buffer. However, if margins are set and the cursor is within the scrolling region, the cursor stops at the bottom margin of the scrolling region.

CURSOR LEFT

Moves the cursor one column to the left.

Host Syntax

EcD

The cursor does not move beyond the leftmost column (Column 1).

This command works just like the ANSI command CUB (CURSOR BACKWARD) with a parameter of 1.

CURSOR RIGHT

Moves the cursor one column to the right.

Host Syntax

EcC

The cursor does not move beyond the rightmost column. If Column mode is set to 132, the cursor may disappear from the screen. This command will not scroll horizontally to keep the cursor in view.

This command works just like the ANSI command CUF (CURSOR FORWARD) with a parameter of 1.

CURSOR TO HOME

Moves the cursor to the home position.

Host Syntax

EcH

The home position is Row 1, Column 1 of the dialog buffer.

CURSOR UP

Moves the cursor up one line without moving it horizontally.

Host Syntax

EcA

The cursor address is based on the first line of the dialog buffer (Row 1, Column 1 is the first position in the buffer), and the cursor stops at the first row of the dialog buffer. However, if margins are set and the cursor is within the scrolling region, the cursor stops at the top margin of the scrolling region.

DIRECT CURSOR ADDRESS

Moves the cursor to the specified line and column.

Host Syntax

```
EcY line column
```

line: specifies the destination line for the cursor. The maximum line range is 96, even if the dialog buffer is larger.

column: specifies the destination column for the cursor. The maximum column range is 80, even if Column mode is set to 132.

NOTE

This command requires that you enter encoded integers as parameter values.

The parameter values for *row* and *column* are ASCII characters that represent the row or column number plus 31.

If a parameter is out of range, the cursor will not change position for that parameter. However, the cursor will move to the other parameter position if it is within the range.

Syntax Example

```
EcY"Sp
```

Moves the cursor to Line 3, Column 1. The ASCII decimal equivalent of " is 34 (3 + 31) and the ASCII decimal equivalent of Sp is 32 (1 + 31).

ENQUIRY

Queries the terminal for its answerback string.

Host Syntax

```
EQ
```

The terminal's answerback string can be set by using the Setup command SET ANSWERBACK STRING, described in Section 5. Your program can use the answerback string to identify the terminal and determine whether the terminal is authorized to use specific programs and data.

You can issue this command from any host command mode. The terminal does not respond to this command in Local mode.

ENTER ALTERNATE KEYPAD MODE

Causes the numeric keypad keys and Function Keys F5 through F8 to assume their Alternate Keypad mode meanings (shown in Table 3-12).

Host Syntax

```
EC =
```

Any other meanings you program into these keys cannot be used as long as the terminal is in Alternate Keypad mode.

Table 3-12 shows the characters transmitted by the numeric keypad keys and function keys as a default and in Alternate Keypad mode. When the terminal is turned on, these keys take on their default meanings.

Table 3-12

ALTERNATE KEYPAD PROGRAMMING CODES^a

Numeric Keypad Key	Characters Sent as Default ^b	Characters Sent in Alternate Keypad Mode
0	0	E _c ?p
1	1	E _c ?q
2	2	E _c ?r
3	3	E _c ?s
4	4	E _c ?t
5	5	E _c ?u
6	6	E _c ?v
7	7	E _c ?w
8	8	E _c ?x
9	9	E _c ?y
-	-	E _c ?m
,	,	E _c ?l
.	.	E _c ?n
ENTER	E _c R	E _c ?M
F5	E _c P	E _c P
F6	E _c Q	E _c Q
F7	E _c R	E _c R
F8	E _c S	E _c S

^a Refer to the discussion *VT52 Mode* earlier in this section for an explanation of how ANSI mode commands affect the codes sent by the keypad and function keys.
^b If these keys are programmed with macros and you haven't disabled key expansion, the terminal sends the macros rather than the characters listed in this column.

ENTER ANSI MODE

Places the terminal in ANSI mode.

Host Syntax

```
E_C<
```

The terminal will interpret all subsequent commands according to ANSI Standard X3.64.

ENTER GRAPHICS MODE

Selects the rulings character set as the G0 character set.

Host Syntax

```
E_C F
```

The terminal will remain in Graphics mode until you issue an EXIT GRAPHICS MODE command. If you issue the ENTER ANSI MODE command while the terminal is still in Graphics mode, the terminal will exit Graphics mode before it exits VT52 mode.

ERASE TO END OF LINE

Erases all characters from the cursor to the end of the current line.

Host Syntax

```
E_C K
```

The cursor position does not change.

This command works like the ANSI command EL (ERASE IN LINE) with a parameter of 0.

ERASE TO END OF SCREEN

Erases all characters from the cursor to the end of the screen.

Host Syntax

```
E_C J
```

The cursor position does not change.

This command works like the ANSI command ED (ERASE IN DISPLAY) with a parameter of 0. It erases text from the cursor position to the end of the dialog buffer, so it makes no difference if margins are set.

EXIT ALTERNATE KEYPAD MODE

Causes the numeric keypad keys and Function Keys F5 through F8 to assume their default meanings, or their programmed meanings if they have been programmed.

Host Syntax

```
E_C >
```

Table 3-12 (under ENTER ALTERNATE KEYPAD MODE) shows the default meanings of the keys.

EXIT GRAPHICS MODE

Restores the G0 character set that was in effect before the current ENTER GRAPHICS MODE command was issued.

Host Syntax

```
E_C G
```

IDENTIFY

Identifies the terminal to the host.

Host Syntax

```
EcZ
```

Report Syntax

```
Ec/Z
```

When the host issues this command, the terminal sends its identifier sequence (E_c/Z) to the host.

REPORT SYNTAX MODE

Sends a Terminal Settings Report that contains the syntax mode status to the host.

Host Syntax

```
Ec#!0
```

Report Syntax

```
%! mode
```

mode: reports the host command mode currently in use at the terminal. Reported as one of the following:

- 0 TEK mode
- 1 ANSI mode
- 2 EDIT mode
- 3 VT52 mode

This command has the same effect as the 4100-style REPORT TERMINAL SETTINGS command issued for the SELECT CODE command (as if E_cIQ%! were sent from the host). See the discussion *Terminal Settings Report* in Section 5 for additional information.

This command is recognized in all host command modes: ANSI, EDIT, TEK, and VT52.

REVERSE LINE FEED

Moves the cursor up one line without affecting the cursor position on the line.

Host Syntax

```
EcI
```

SELECT CODE

Causes the terminal to recognize ANSI, TEK (4100-style), or VT52 command syntax. Also used to select EDIT mode.

Host Syntax

```
Ec%! syntax
```

Setup Syntax

```
CODE syntax
```

syntax: specifies the host command mode in which you want to operate:

Host	Setup	
0	TEK	Selects TEK mode
1	ANSI	Selects ANSI mode
2	EDIT	Selects EDIT mode
3	VT52	Selects VT52 mode

Defaults: Factory = TEK
Omitted = TEK

The syntax of TEK, ANSI, and VT52 commands are not compatible. If you are using commands from the host in one mode and want to execute one or more commands in another mode, you must issue the Select Code command with the appropriate parameter.

EDIT mode allows the terminal to be used with VT100 application programs. See the discussion of EDIT mode at the beginning of this section.

This command is recognized in all host command modes: ANSI, EDIT, TEK, and VT52.

Section 4

GRAPHICS CONCEPTS

This section explains the concepts behind the terminal's graphics features and introduces the graphics commands.

Reading this section in its entirety will help you understand how commands work together to control the display and create graphics. Once you understand the general nature of the commands, refer to Section 5 for detailed descriptions of each command.

The topics in this section include:

- *Using Graphics Commands*
- *Displaying Dialog Between a Host and a User*
- *Understanding the Graphics Display and Graphics Memory*
- *Displaying Colors*
- *Creating Images With Graphics Primitives*
- *Creating Images With Pixel Operations*
- *Using 4010 GIN*
- *Using Macros*
- *Putting Together a Graphics Program*

Each discussion concludes with a table summarizing the commands that control the features described in that discussion.

COMMAND HINTS

- Commands are always identified in this manual by a descriptive name in uppercase letters.
- The terminal has three command sets and four host command modes. ANSI and EDIT modes use the ANSI command set; VT52 mode uses the VT52 command set; and TEK mode uses the 4100 command set.
- A program can freely switch command modes to access the terminal's full feature set (use the SELECT CODE command, which works in all modes).
- You can issue commands from a host program (using host syntax) or from the keyboard (using Setup syntax); the host and Setup versions of a command do exactly the same thing.
- When you select Setup (by pressing the Setup key), you can issue Setup commands from the keyboard without regard to the host command mode. Setup syntax uses simple keywords and ordinary integers.
- When you issue 4100-style commands from the host, the terminal must be in TEK mode and you must encode parameter values. The different parameter types (and their encoding schemes) are explained in Section 5. Section 6 contains sample routines for encoding these parameters.
- You can save many command settings in the terminal's nonvolatile memory by issuing the SAVE NONVOLATILE PARAMETERS command; then your terminal's settings will be appropriate for your application every time you turn on the terminal.
- 4100-style commands are described at the end of Section 5. ANSI and VT52 commands are described in Section 3. Each command is described in detail, listed alphabetically by command name.
- 4100-style reports are described at the end of Section 5. ANSI and VT52 reports are described alphabetically with the command descriptions in Section 3. Section 6 contains sample routines for decoding reports.
- If you're looking for information on a specific topic, try each section's detailed Table of Contents and functional listing of commands — you'll find these on each section divider. Also try the general Table of Contents and the Index.

USING GRAPHICS COMMANDS

The terminal's graphics commands are a subset of Tektronix 4100-style commands. Most 4100-style commands have both *host syntax* and *Setup syntax*:

- *Host syntax* typically consists of an *escape sequence*, usually followed by one or more parameters. The escape sequence consists of the escape character (Esc), followed by two identifying characters (an *opcode*). Your program must encode most parameters used in host commands.
- *Setup syntax* consists of an alphabetic word that identifies the command's purpose (like, *MACROSTATUS*, which displays the contents of a macro), usually followed by one or more parameters. Setup commands are entered directly from the terminal keyboard, using integer or keyword parameters.

NOTE

*Before issuing TEK mode (or, 4100-style) commands from the host, the terminal must be in TEK mode — issue the SELECT CODE command ($\text{Esc}\%!\text{0}$ in host syntax — **CODE TEK** in Setup syntax).*

Remember that TEK mode is a host command mode — when commands are entered from the keyboard, the terminal accepts any command that has Setup syntax, regardless of the host command mode.

When you issue 4100-style commands from the host, be sure the terminal is in *TEK mode*. To issue 4100-style commands directly from the keyboard, press the Setup key. Remember that while the terminal is in Setup, you can issue *any* command that has Setup syntax from the keyboard, regardless of the mode that the host last placed the terminal in.

When you select TEK mode after using EDIT mode, the terminal settings may not be appropriate. This is because EDIT mode resets a number of terminal characteristics to emulate a VT100 terminal. Therefore, you may need to reset them so your program will work as expected. See the command description for SELECT CODE in this section for details.

You can customize your terminal by saving the settings of many commands in *nonvolatile memory*. The terminal retains these settings even when it's turned off. Save settings by using the SAVE NONVOLATILE PARAMETERS command. The command descriptions in Section 5 indicate the commands whose settings you can save in nonvolatile memory.

NOTE

You can use host syntax from the keyboard while in Local mode, which can be useful for debugging. See the discussion Using Host Syntax From the Keyboard in Section 2 and the command description for the LOCAL command in Section 5.

Throughout this manual, command names are shown in full uppercase letters (for example, DEFINE MACRO), and parameter names are shown hyphenated and in italics (like this — *macro-number*).

DISPLAYING DIALOG BETWEEN A HOST AND A USER

In writing a graphics program, you might want to build in user prompts — perhaps to give next-step instructions or to request data. This discussion provides the information you need to do so.

DISPLAY AREAS

The terminal can display two types of information — (1) graphics and (2) dialog between the host program and a user. By setting up a *dialog area*, your program can display host messages and user responses without interfering with the graphics, which is always displayed in the *graphics area*.

You can display the graphics area and the dialog area separately or simultaneously. Displaying them simultaneously allows users to see host prompts while they're working with their graphics display. And if you make the dialog area background *transparent*, it won't obscure the graphics behind it.

Alphatext

The text that the terminal uses for displaying messages in the dialog area is called *alphatext*. Any characters sent from the host that are not parts of commands are displayed as alphatext. Alphatext is displayed in the dialog area, unless the dialog area has been disabled (see *Emulating 4010 Series Terminals*, later in this section).

Alphatext characters are 5 by 7 pixels in size, displayed within a 6 by 12 pixel *character cell*.

NOTE

Be sure the terminal is in Alpha mode when displaying messages in the dialog area — use the ENTER ALPHA MODE command (\cup s).

The terminal must be in *Alpha mode* to display alphatext. Alpha mode is one of three *implicit command modes* — discussed later in this section under *Creating Images With Graphics Primitives*. The terminal powers up in Alpha mode. Any time your program specifies either of the other two implicit command modes (Vector and Marker mode), be sure you issue \cup s (the ENTER ALPHA MODE command) before sending messages to the dialog area.

CONTROLLING THE DIALOG AREA AND DIALOG AREA BUFFER

The terminal stores lines of dialog in the *dialog area buffer* (or, simply *dialog buffer*), a part of program memory set aside for that purpose. The dialog area is the part of the dialog buffer that appears on the screen.

The factory default dialog area is 30 lines, and the factory default dialog buffer size is 49 lines. The dialog area can vary in size from two lines (at the bottom of the screen) to 30 lines (occupying the entire screen). You can make the dialog buffer as large as program memory permits.

When lines of dialog accumulate, the lines at the top disappear from (or, scroll off the top of) the dialog area. If the dialog buffer is larger than the dialog area, the lines that scroll out of the dialog area are retained in the dialog buffer, and you can scroll the buffer (use the Joydisk) to bring parts of it into view. Once the buffer fills up, however, the terminal discards the oldest line for every new line.

To change the dialog area size, use the SET DIALOG AREA LINES command; use the SET DIALOG AREA BUFFER SIZE command to change the size of the buffer.

Figure 4-1 shows a small dialog area superimposed on the graphics area of the terminal screen. You can see that the dialog buffer is quite large, but the dialog area displays only a few of those lines.

Displaying Messages in the Dialog Area

To display messages in the dialog area, it must be enabled *and* visible. If the dialog area is enabled but *invisible*, dialog accumulates in the dialog buffer and is not displayed until you make the dialog area visible.

Use the ENABLE DIALOG AREA command to enable the dialog area. To make the dialog area visible or invisible, use either the SET DIALOG AREA VISIBILITY command or the Dialog key on the keyboard. You can issue SET DIALOG AREA VISIBILITY either immediately following the ENABLE DIALOG AREA command or whenever you want to display stored dialog.

NOTE

To emulate a Tektronix 4010 Series terminal (which doesn't have a dialog area), you can disable the dialog area. Then, alphatext will be displayed in the graphics area. See the discussions Emulating 4010 Series Terminals and Using Alphatext in Graphics, later in this section.

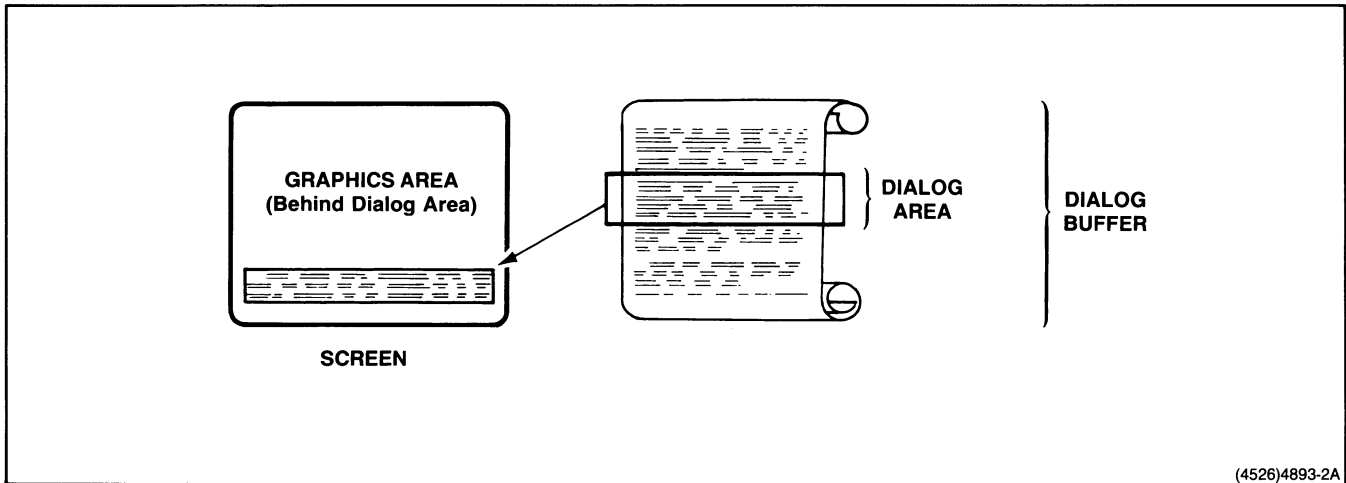


Figure 4-1. The Dialog Area and the Dialog Area Buffer.

Using Colors in the Dialog Area

You might want to specify dialog area colors that will clearly distinguish the dialog area from the graphics area — or you might want to specify colors that will allow users to see through the dialog area, keeping it from obscuring the graphics behind it. Or, you might want to specify colors simply to create an aesthetically pleasing display. The SET DIALOG AREA INDEX commands allows you to assign colors that will achieve any of these effects.

The SET DIALOG AREA INDEX command assigns three colors:

- *Character color* — The color for text in the dialog area
- *Character cell color* — The color of the cell surrounding each character
- *Background color* — The color of the dialog area before anything is written on it

Figure 4-2 shows where these colors are displayed in the dialog area.

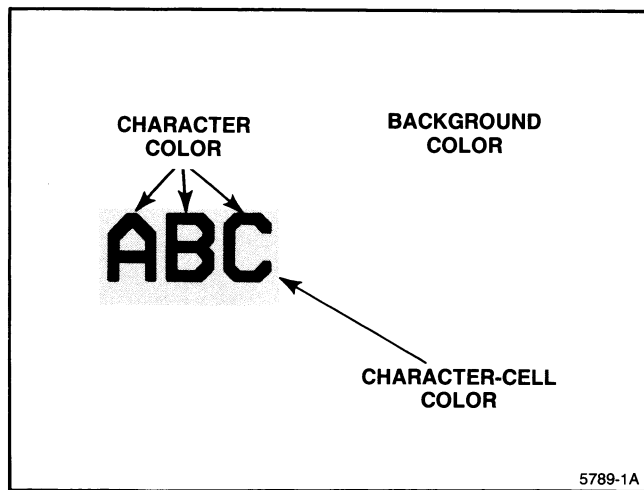


Figure 4-2. How Colors Are Displayed in the Dialog Area.

To allow users to see graphics behind the dialog area, make both the dialog background and the character background *transparent*. This makes alphanumerics look as though it were written on a piece of glass in front of the graphics.

For more information on transparency, see *Displaying Colors* later in this section.

You can use the ANSI command SGR (SET GRAPHICS RENDITION) to select blinking characters in the dialog area. See the discussion *Selecting Underscored or Blinking Text* in Section 3 for a more complete discussion of the SGR command.

BUILDING USER PROMPTS INTO A PROGRAM: AN EXAMPLE

Here's one way to set up your program to display user prompts without cluttering up the graphics area. First, create a dialog area that is six lines long. Then enable the dialog area (so that it stores dialog in the dialog buffer), but don't make it visible — until you are ready to display a prompt to the user. At any point in the program, you can now make the dialog area visible, display a prompt, and wait for the user's response. Once the user responds, make the dialog area invisible again until you are ready to display another prompt.

To use this scheme, you need these three commands:

- SET DIALOG AREA LINES
- ENABLE DIALOG AREA
- SET DIALOG AREA VISIBILITY

EMULATING 4010 SERIES TERMINALS

You can emulate 4010 Series terminals, which do not have dialog area capability, by disabling the dialog area. When you send alphatext to a terminal whose dialog area is disabled, the alphatext is displayed in the graphics area.

However, alphatext display is limited. You can't rotate, slant, or size it, as you can with graphtext. The discussion *Using Alphatext in Graphics* describes the attributes you can set for alphatext that appears in the graphics area.

NOTE

Alphatext in graphics is provided for compatibility with older terminals. For this terminal, it's better to use graphtext in the graphics area and alphatext only in the dialog area (unless you're emulating a 4010 Series terminal).

DIALOG AREA COMMANDS

Table 4-1 summarizes the commands that control the dialog area. Most of these commands can either be sent from the host or entered at the keyboard through Setup.

Besides the settings made with 4100-style commands listed in Table 4-1, there are other dialog area and keyboard characteristics that you can set using ANSI-style commands. These characteristics stay in effect in all modes. See the discussion titled *Controlling the Dialog Display and the Keyboard* in Section 3 for information about these characteristics and the commands that set them.

NOTE

Besides the 4100-style commands listed in Table 4-1, there are ANSI-style commands that also set terminal characteristics. See Section 3 for information about these commands.

Table 4-1
DIALOG AREA COMMANDS

Host Command	Setup Command	Function
CLEAR DIALOG SCROLL	CLEARDIALOG	Erases the dialog buffer
ENABLE DIALOG AREA	DAENABLE	Enables or disables the dialog area
SET ALPHA CURSOR INDICES	ACURSOR	Assigns color indices to the alpha cursor
SET DIALOG AREA COLOR MAP	DACMAP	Specifies the colors assigned to color indices in the dialog area
SET DIALOG AREA BUFFER SIZE	DABUFFER	Specifies the number of lines available for storing text in the dialog buffer
SET DIALOG AREA INDEX	DAINDEX	Specifies the color indices for alphatext characters, character cell background, and dialog area background
SET DIALOG AREA LINES	DALINES	Specifies the number of lines of the dialog area that is displayed on the screen
SET DIALOG AREA VISIBILITY	DAVISIBILITY	Makes dialog area visible or invisible
SET DIALOG AREA WRITING MODE	DAMODE	Changes the function of the Space and Underscore characters so that the Underscore character underscores an existing character and the Space character just moves the cursor to the right without erasing the existing character

UNDERSTANDING THE GRAPHICS DISPLAY AND GRAPHICS MEMORY

This discussion explains how the terminal displays graphics images on its screen and how it builds them in its graphics memory. This discussion introduces terms (*pixel*, *color index*, *graphics memory*, and *terminal space*) that are used in subsequent discussions *Displaying Color*, *Creating Graphics Images With Graphics Primitives*, and *Creating Graphics Images With Pixel Operations*.

UNDERSTANDING THE GRAPHICS DISPLAY

The terminal uses a *raster display system* to display graphics, in much the same way as a television displays a picture. In a raster display system, an electron beam inside the terminal (or television) illuminates points (called *pixels*) on the screen to create a display.

The smallest screen element that the terminal can address is called a *pixel*. The terminal draws each individual pixel as a mixture of red, green, and blue, creating a single color. When you look at the terminal display, your eyes blend the pixels and give the illusion of a continuous form.

Figure 4-3 shows how pixels are arranged to give the illusion of a straight diagonal line.

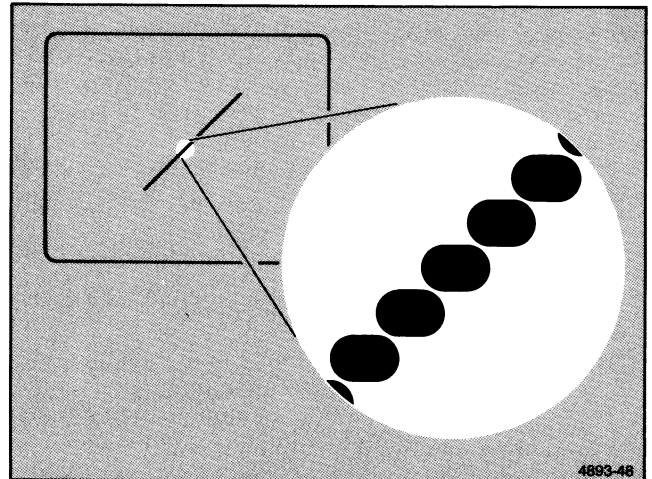


Figure 4-3. Magnified View of Pixels in a Line.

GRAPHICS CONCEPTS
GRAPHICS DISPLAY & MEMORY

The terminal stores the color for each pixel in graphics memory as a number, which is called a *color index* because it is an index into a special part of program memory called the *color map*. Figure 4-4 shows how the terminal (1) reads color indices for each pixel from graphics memory, (2) looks in the color map for the entry corresponding to that index, and (3) displays each pixel on the screen as part of its raster display.

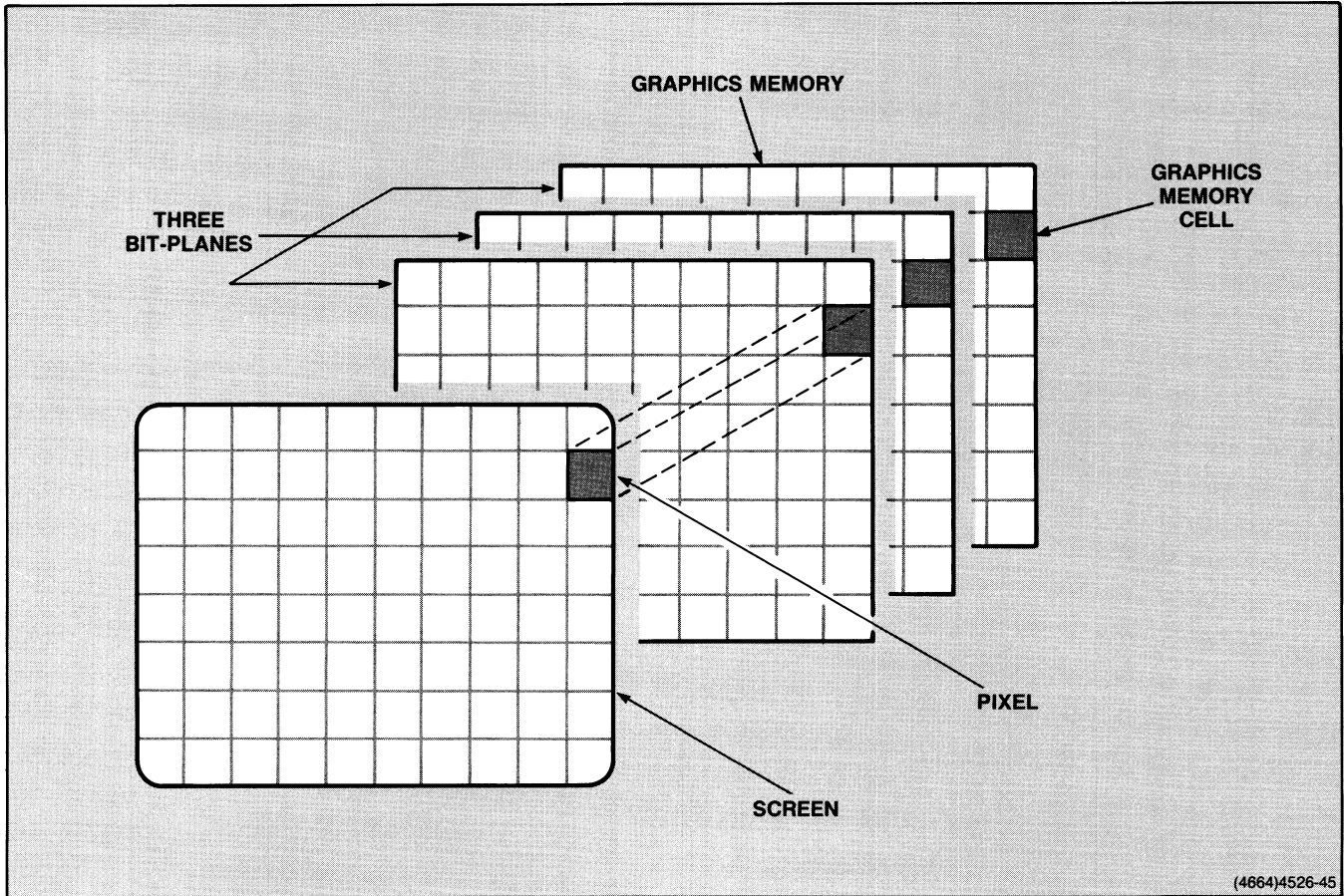


Figure 4-4. How Colors Map to the Screen.

GRAPHICS MEMORY

Graphics memory is a special part of memory the terminal uses to store graphics images for display on the terminal screen (graphics memory corresponds to the bit planes shown in Figures 4-4 and 4-5). The graphics memory for your terminal is 512 by 360 pixels; it is divided into *on-screen memory* (dimensions 480 by 360 pixels) and *off-screen memory* (dimensions 32 by 360 pixels).

Each location in *on-screen memory* is paired with a pixel on the screen. *Off-screen memory* is an area of graphics memory in which you can create and store pixel images that you will later copy into the on-screen memory with the `PIXEL COPY` command.

You can visualize graphics memory as a three-dimensional array of bits. The height and width of the displayed part of this array correspond to the dimensions of the screen in pixels. The depth of the screen in bits is the number of *bit planes*; your terminal has three bit planes.

Figure 4-5 shows how screen pixels correspond to the graphics memory locations.

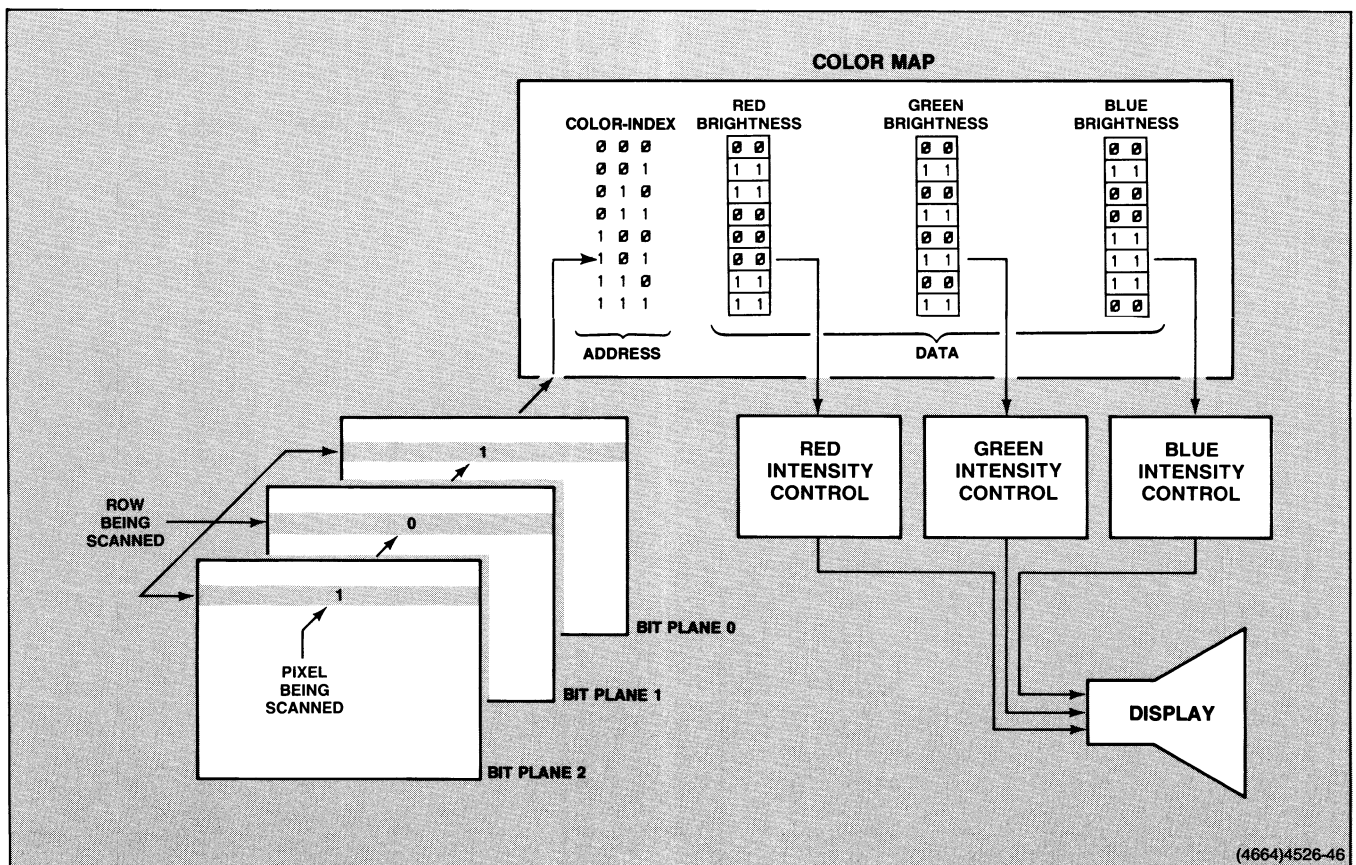


Figure 4-5. Screen Pixels and Graphics Memory.

TERMINAL SPACE

Terminal space is the imaginary plane defined by the coordinate system you use when issuing graphics primitive commands.

When you issue graphics primitive commands, you specify locations as *xy*-coordinates. The maximum size for each coordinate is determined by the terminal's addressing limits. The terminal's addressable resolution is 4096 by 4096 *terminal space units*, thus the *x*- and *y*-coordinates can range from 0 to 4095.

Mapping Terminal Space to the Screen

The screen can show all or part of terminal space. By default, your terminal displays a 4096 by 3072 *window* into terminal space, with the lower-left corner of the screen corresponding to terminal space coordinates 0,0 and the upper-right corner to terminal space coordinates 4096,3072. The terminal-space data that falls within the window's boundaries makes up the image on your display screen.

Figure 4-6 illustrates the relationship between terminal space and the default window.

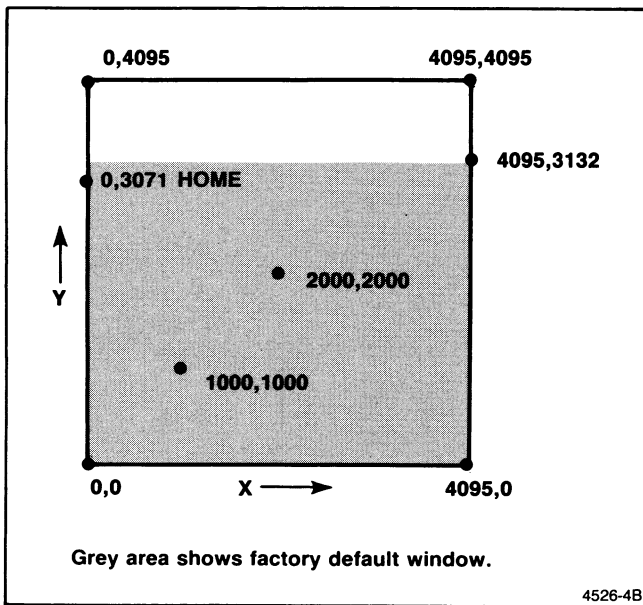


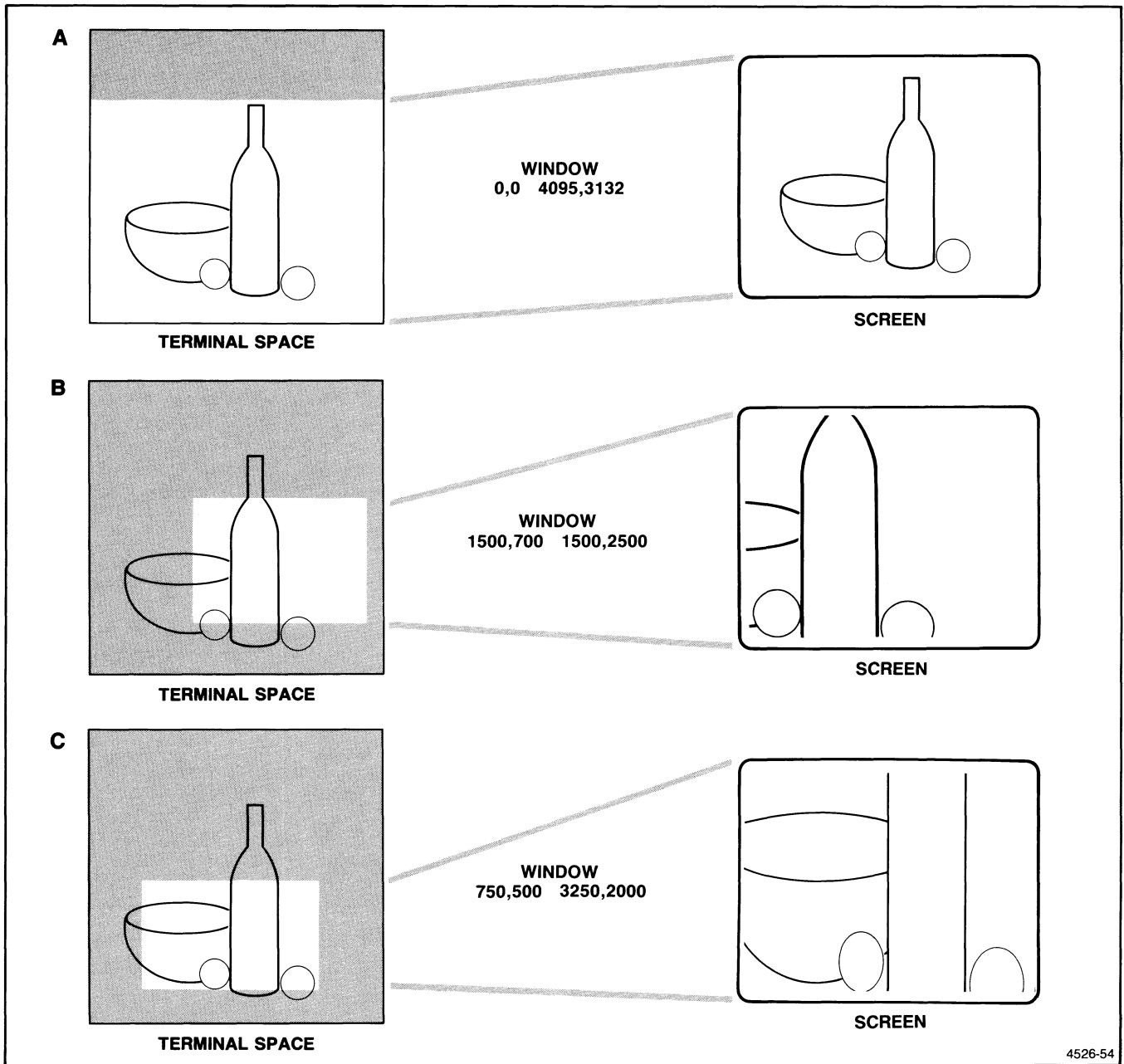
Figure 4-6. Terminal Space and the Default Window.

You can use the `SET WINDOW` command to change the window's size and position, thus controlling the part of terminal space that you see.

Graphics images that lie within the bounds of the window are mapped onto the screen. The graphics images within the window are scaled to fill the screen, and any parts of the image that fall outside the window are *clipped* (truncated).

You can distort the image in the viewport by making the aspect ratio of the window different from that of the screen. However, if you set either the height or the width of the window to zero, the terminal will automatically set the aspect ratio of the window equal to the aspect ratio of the screen.

Figure 4-7 shows how three windows in terminal space map to the screen. Figure 4-7a shows the default window. In Figure 4-7b, the width of the window is specified as zero, so the terminal has set the window width to a value that results in an undistorted image. Figure 4-7c shows how selecting a window with different proportions than the screen results in a distorted image.



4526-54

Figure 4-7. How Windows Map Terminal Space to the Screen.

DISPLAYING COLORS

This discussion explains how the color map assigns colors to color indices and how you can change the colors in the color map.

USING COLOR INDICES

You can specify colors for the lines, panels, and text that you use to build a graphics image. You assign a color to part of an image by using a number called a *color index*. There are eight color indices available in the graphics area, and eight additional indices are available for the dialog area. This means that the terminal can display up to eight colors in the graphics area and eight colors in the dialog area simultaneously.

Each color index is an integer that functions as a pointer into the *color map*, an area of program memory that holds the color definition for each index.

The terminal uses the color map to translate an index number into a color on the display. For example, the terminal's factory default defines Index 2 as a color mixture that creates red. So, anything you assign to Index 2 is displayed in red. If you then change the color mixture for Index 2 to blue, anything displayed in Index 2 will be blue.

Even if you have assigned a color mixture to Index 0 in the dialog area, Index 0 always means *transparent* when it's used as the character-cell or dialog background color. When it's used for the character color, Index 0, like other color indices, is a particular color mixture.

The factory default colors for the graphics area are the same as the dialog area:

0 = Black	4 = Blue
1 = White	5 = Cyan
2 = Red	6 = Magenta
3 = Green	7 = Yellow

Erase Index. You can specify the *erase index* (or wipe index) for the graphics area with the SET VIEW ATTRIBUTES command. The erase index is the color to which the graphics area background is set when you erase it.

SPECIFYING COLORS FOR THE COLOR MAP

You specify colors for the terminal's color map, by giving numeric values for the HLS coordinates (*hue*, *lightness*, and *saturation*).

You can begin to understand the HLS system by looking at the color cone in Figure 4-8. Each color is a point within the volume of the double-ended cone defined by three vectors: hue (which is defined as an angle), and lightness, and saturation (which are defined as horizontal and vertical displacement). Appendix E shows the HLS system color cone in color.

Hue is the basic sensation we think of as color. In the color cone, it is the angle formed by rotating a vector around the axis of the cone, with blue as the reference. A hue of 0° (or 360°) corresponds to blue, 120° to red, and 240° to green, with intermediate shades corresponding to intermediate rotations. You specify hue as an integer representing degrees in the range -32768 to +32767. The terminal converts integers less than 0 or more than 359 to values in the range 0 to 359 by a modulo function.

Lightness is how bright or dull a color appears — that is, how much light is emitted by a color. In the color cone, lightness is determined by the position of a vector along the axis of the cone. A lightness of 0% is black, and a lightness of 100% is white. (At lightness 0% or 100%, saturation and hue are irrelevant.) You specify lightness with an integer representing percentage in the range 0 to 100.

Saturation is the intensity of a color. In the color cone, it is the radial distance of the vector from the cone axis. A saturated color is very intense, while a less saturated color is one that appears grayed or muted. A saturation of 0% is simply a shade of gray, while a saturation of 100% gives the most intense possible color having that hue and brightness. You specify saturation as an integer representing percentage in the range 0 to 100.

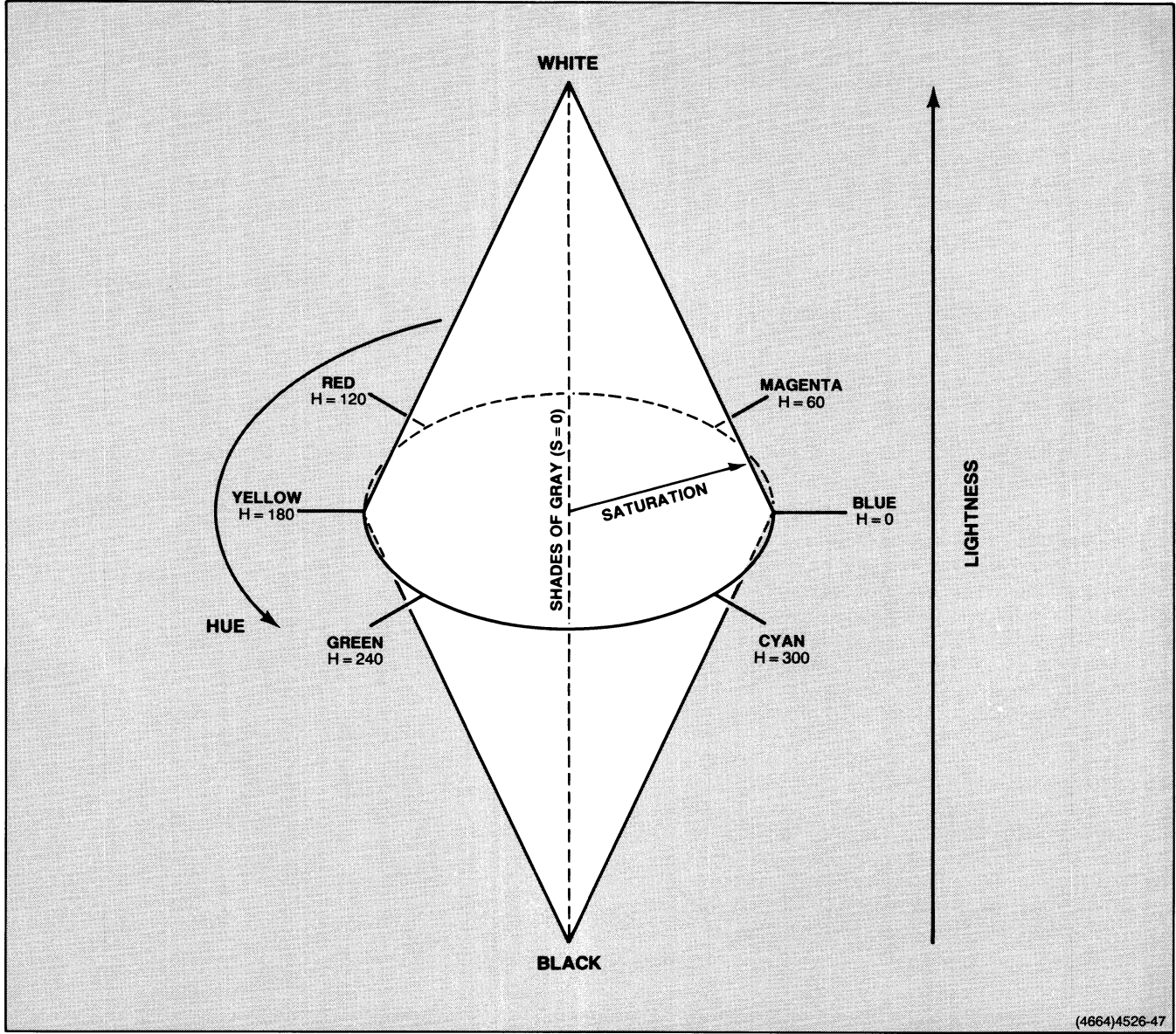


Figure 4-8. The HLS System Color Cone.

CHANGING THE COLOR MAP

You can define new color mixtures using the HLS color coordinate system and assign them to color indices. You can assign colors from the host or from the keyboard. For example, you can assign colors in the graphics area with the SET SURFACE COLOR MAP command (in Setup, CMAP), and in the dialog area with the SET DIALOG AREA COLOR MAP (in Setup, DACMAP).

NOTE

The user can change the color mixtures in the dialog area color map from the keyboard by using the Interactive Color Interface. So, if it is important to your program to have specific colors assigned to particular indices, you will need to reissue the SET SURFACE COLOR MAP or SET DIALOG AREA COLOR MAP command to ensure that you get the colors you want.

You can also alter colors from the terminal keyboard by using the *Interactive Color Interface*, which is described in the Operators Manual for that terminal. You select the Interactive Color Interface display by pressing the Menu key. This displays the crosshair cursor, the index number and surface number associated with the crosshair cursor position, and a banner showing function key labels along with the HLS coordinate values of the color index displayed at the current cursor position.

You can change the hue, lightness, or saturation of the color index (everywhere it appears on the screen) by positioning the cursor over an area of color and pressing Function Keys F1 through F3 respectively. Or, you can hold down Function Key F5 to display a color menu and then move the cursor to a position on the menu to select a color by name. Pressing the F4 key will reset the color map to its original value.

Defining Color Mixtures

The SET SURFACE COLOR MAP and SET DIALOG AREA COLOR MAP commands define color mixtures by sending a *quadruple* for each color index being defined (you can change more than one at a time).

Each quadruple consists of an index number and the three color coordinates that define the color mixture for that index. For example, to define the color mixture for Index 1 in the dialog area as red instead of white (Index 1's default color), you could send the SET DIALOG AREA COLOR MAP command and, using the HLS system, specify these four values in the quadruple: 1, 120, 50, and 100 (that is, Index 1, hue 120°, lightness 50%, and saturation 100%).

When you define color mixtures, you can select from the terminal's total range of 64 colors. Because the HLS color system covers the total color spectrum, the terminal assigns a range of HLS values to the same color. For example, at 20% lightness and 75% saturation, the range of hues 345° to 15° map to the same value of *blue*.

Figure 4-9 shows how changing color indices can affect the appearance of a display by changing the color map.

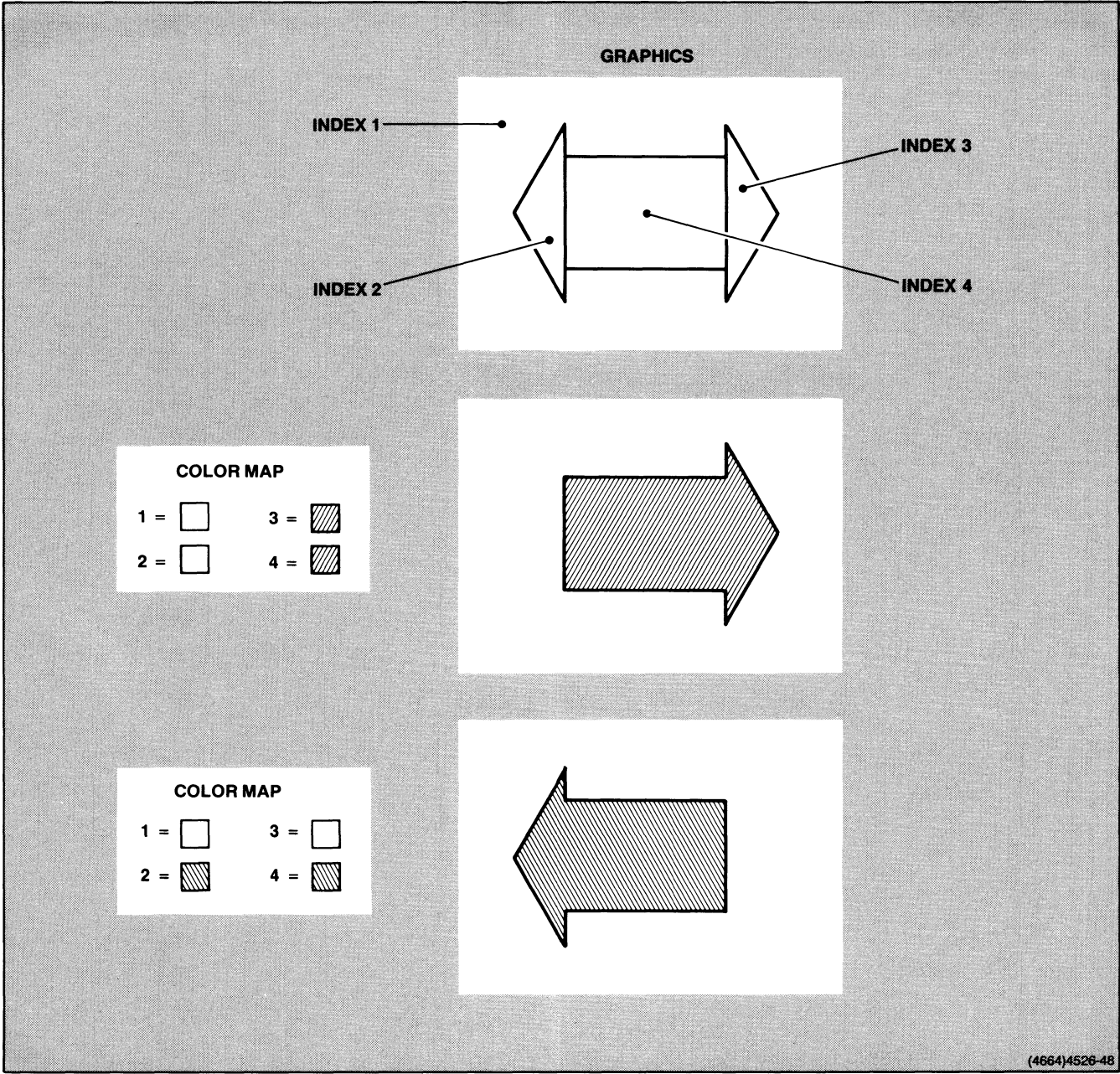


Figure 4-9. The Effect of Changing the Color Map.

COLOR COMMANDS

Table 4-2 summarizes the commands used for changing and defining colors.

Table 4-2
COLOR COMMANDS

Command Name	Setup Name	Function
SELECT FILL PATTERN	FILLPATTERN	Selects a color or predefined fill pattern to fill a panel
SET ALPHA CURSOR INDICES	ACURSOR	Assigns two color indices to the alpha cursor
SET DIALOG AREA COLOR MAP	DACMAP	Specifies the colors assigned to color indices in the dialog area
SET DIALOG AREA INDEX	DAINDEX	Specifies the color index for alphanext characters, character-cell background, and dialog area background
SET LINE INDEX	LINEINDEX	Specifies the color index for all subsequent lines, panel boundaries, and markers
SET SURFACE COLOR MAP	CMAP	Defines the color map for the graphics area
SET TEXT INDEX	GTINDEX	Specifies the color index for all text displayed in the graphics area

CREATING IMAGES WITH GRAPHICS PRIMITIVES

Graphics primitives are the fundamental units of a graphics display — that is, the basic building blocks from which the graphics display is constructed. The terminal's graphics primitives are:

- Vectors (lines)
- Markers (predefined symbols)
- Text (graphtext and alphatext in the graphics area)
- Panels (closed regions)

Each graphics primitive is drawn in response to a *graphics primitive command*. These commands define primitive attributes (such as the size and position of markers and text), the coordinates of vector endpoints, and the shape of panels. Other commands specify color attributes: the color and line style of vectors, the fill color or pattern for panels, and so on.

The terminal draws graphics primitives starting from the *graphics position*, which is the point in terminal space at which the last graphics operation left off. The graphics position becomes the starting point that the terminal uses when executing the next graphics primitive command.

The terminal offers two methods for issuing graphics primitive commands: *explicit* and *implicit*.

IMPLICIT COMMAND MODES

You can send any command *explicitly* as a command code followed by the appropriate parameters. You can also send some commands *implicitly* — that is, without issuing the command code; you need only put the terminal in one of its three *implicit command modes*: Alpha mode, Marker mode, and Vector mode. The terminal must be in TEK mode (a host command mode) to enter any of these implicit command modes.

In these modes, the terminal implicitly interprets data sent from the host:

- In Alpha mode, the terminal interprets incoming characters as alphatext. (This is the only means for displaying alphatext.)
- In Vector mode, the terminal interprets incoming characters as encoded xy-coordinates to be executed as MOVE and DRAW commands (the first coordinate is interpreted as a MOVE, the rest are interpreted as DRAWs).
- In Marker mode, the terminal interprets incoming characters as encoded xy-coordinates to be executed as DRAW MARKER commands.

By freeing you of the necessity of explicitly sending the escape sequence for these commands, Vector and Marker modes reduce communication traffic from the host to the terminal.

Hint. Several commands and keyboard actions will put the terminal in Alpha mode (thus canceling Vector mode or Marker mode), so you must keep track of how your program uses these commands and reissue ENTER VECTOR MODE or ENTER MARKER MODE as needed. If you are in Vector or Marker mode, issuing the PAGE, RESET, or ENABLE 4010 GIN command will put the terminal in Alpha mode. If the dialog area is disabled, pressing the G Eras key or Return key also puts the terminal in Alpha mode.

VECTORS

A *vector* is simply a straight line drawn between two points in terminal space. The current graphics position defines the starting point of the vector; the position sent as the parameter of the DRAW command defines the vector's endpoint.

To display a vector, move the graphics position with a MOVE command to the vector's starting point, then draw the vector with a DRAW command. As pointed out earlier, you can explicitly send the MOVE and DRAW commands to create vectors, or you can enter Vector mode and implicitly send just the xy-coordinate pairs. Figure 4-10 shows both methods for drawing and displaying a line.

You can simulate curves with a series of short vectors.

Line Attributes

Before drawing vectors, you can select the *line attributes* (style and color) to control how the line will be displayed. Line attributes remain set until you change them.

You can choose one of eight different line styles with the SET LINE STYLE command. Figure 4-11 shows the terminal's predefined line styles.

You can use the SET LINE INDEX command to select the color that the terminal uses to draw vectors and markers.

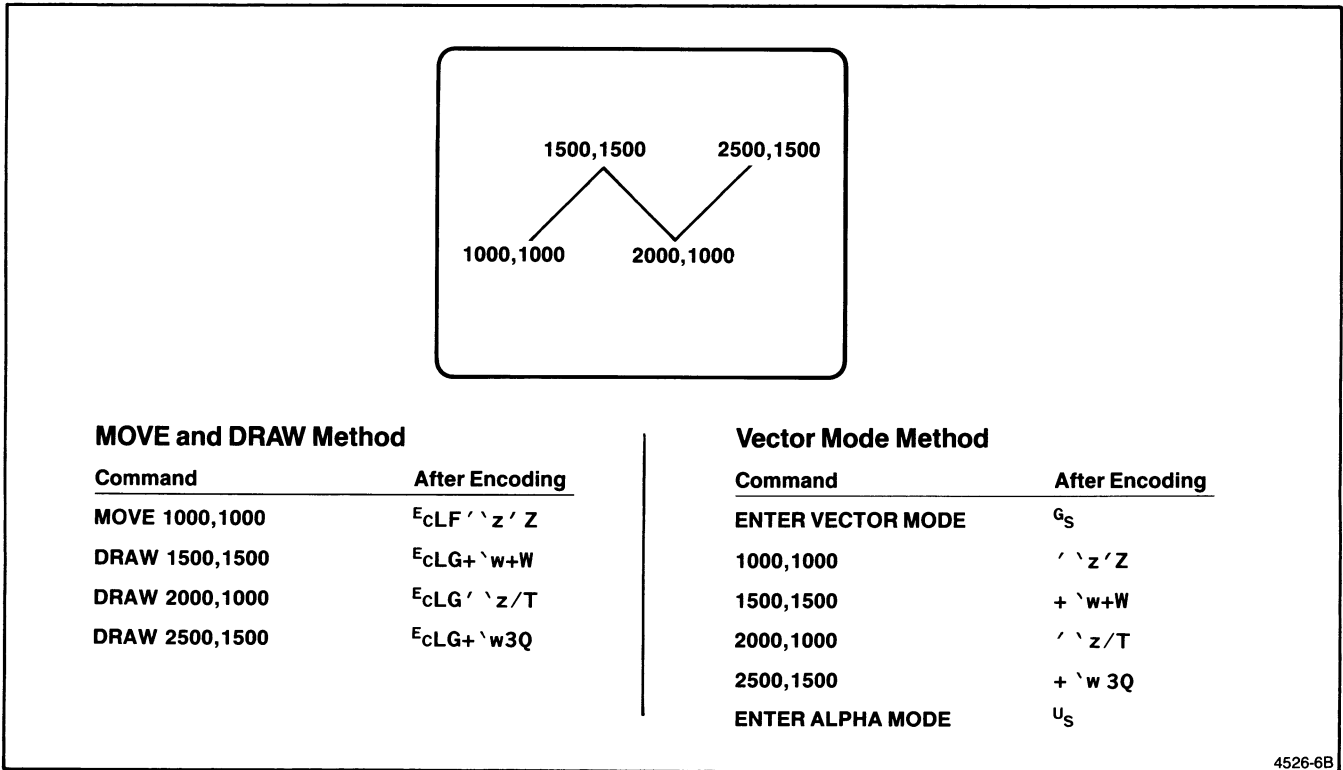


Figure 4-10. Two Methods for Displaying a Line.

Parameter	Line Style
0	—————
1
2	- - - - -
3	- - - - -
4	- - - - -
5
6	- - - - -
7	- - - - -

4526-19A

Figure 4-11. Line Styles.

Using MOVE and DRAW to Draw Lines

The MOVE-and-DRAW method of creating lines uses explicit command syntax — that is, you must issue the entire command for each MOVE or DRAW.

The MOVE command sets the graphics position, but does not draw a line to that position. This command's effect is analogous to lifting a pen from a drawing and moving it to a new location.

The DRAW command draws a line from the graphics position to the position specified in the DRAW command. The position specified in the command becomes the new graphics position.

Using Vector Mode to Draw Lines

You can put the terminal in Vector mode by sending the ENTER VECTOR MODE command (which is the control character G_s). When in this mode, the terminal interprets all characters in the range S_P through D_T as xy-coordinates. (See the description of *xy-coordinate* in Section 5 for instructions on encoding xy-coordinates.)

In Vector mode, the first characters directly following the G_s character are treated as an xy-coordinate and invoke an implicit MOVE to that location. If you want the first implicit command after the ENTER VECTOR MODE command to be a DRAW rather than a MOVE, send the sequence $G_s B_L$. The terminal bell will ring, and the following xy-coordinate will be an implicit DRAW.

Subsequent coordinates are implicit DRAWs, each drawing a line from the current graphics position to the new graphics position specified by the coordinates. If you want to start a line at a new position without drawing a line to that position, just send another ENTER VECTOR MODE command. The next line is drawn from the next position you specify.

The terminal must be in Alpha mode (which is the default implicit command mode) to enter Vector mode. The terminal ignores the G_s character in Marker mode.

You can exit Vector mode by entering either Alpha or Marker mode. There are other ways to exit Vector mode; see the hint under the discussion *Implicit Command Modes* earlier in this section.

VECTOR COMMANDS

Table 4-3 summarizes the commands used for displaying vectors.

Table 4-3
VECTOR COMMANDS

Command Name	Setup Name	Function
DRAW	DRAW	Draws a vector from the current graphics position to a new position
ENTER VECTOR MODE	(none)	Puts the terminal in Vector mode
MOVE	MOVE	Moves the current graphics position without drawing a vector
SET 4014 LINE STYLE	(none)	Specifies line styles compatible with Tektronix 4014, 4016, and 4114 terminals
SET LINE INDEX	LINEINDEX	Specifies the color index for all subsequent lines, panel boundaries, and markers
SET LINE STYLE	LINESTYLE	Specifies the line style for subsequent lines and panel boundaries

MARKERS

A *marker* is a predefined symbol drawn at a given coordinate. You can use markers anywhere that you need to identify points on a display — perhaps to identify towns on a map, important points on a graph, or registration points on graphics overlays.

Before displaying a marker, you can choose any of eleven marker types with the SET MARKER TYPE command. The default marker is a dot. Figure 4-12 shows the terminal's predefined marker types.

When you issue the DRAW MARKER command to specify where to draw the marker, the terminal displays the marker at the indicated location, using the marker type selected in the most recent SET MARKER TYPE command. The SET LINE INDEX command controls the color of the marker.

A more efficient way to put markers in a drawing is to use Marker mode, especially if you need to use a lot of markers. Since you send only the xy-coordinates, you substantially reduce the data communication time and expense.

The terminal draws markers with solid lines. The size of the marker on the screen is independent of the current window.

Parameter	Marker Type	Parameter	Marker Type
0	·	6	□
1	+	7	◇
2	+	8	▣
3	*	9	◇
4	○	10	▣
5	×		

4526-42A

Figure 4-12. Marker Types

MARKER COMMANDS

Table 4-4 summarizes the three commands for displaying markers.

Table 4-4
MARKER COMMANDS

Command Name	Setup Name	Function
DRAW MARKER	MARKER	Draws a marker at a specified location
ENTER MARKER MODE	(none)	Puts the terminal in Marker mode
SET MARKER TYPE	MARKERTYPE	Specifies the kind of marker to be drawn

DISPLAYING TEXT IN THE GRAPHICS AREA

*Graph*text is a special type of text designed for use in the graphics area. Although you can use *alphat*ext in the graphics area, *graph*text is more versatile — you can resize and rotate *graph*text, and you can display it in different directions.

You can use any printable characters as *graph*text, including the ASCII characters `SP` through `~` (ADE 32 through 126). The terminal detects an error for characters outside this range.

Default *graph*text characters come from the *alphat*ext character sets. When you use these characters as *graph*text, you can rotate and size the characters, and set their *character path*; character path is the direction each character is placed in relation to the previous character — that is, stacked on top of or below each other, or written to the right or left of each other.

The default character set for your terminal is determined by the keyboard — the ASCII/North American keyboard selects the ASCII/North American character set, for example. Other character sets are listed in Appendix A. You can use the ANSI command `SCS` (SELECT CHARACTER SET) command, described in Section 3, to make one or two of these character sets available to your program or to the user. Then you can use 4100-style `SET ALPHATEXT FONT` command or the ANSI commands `SI` (SHIFT IN) and `SO` (SHIFT OUT) to invoke either of the character sets previously selected with the `SCS` command.

Before you send the `GRAPHIC TEXT` command — the command that displays *graph*text — you can specify several attributes that determine how it is displayed (see Figure 4-13, which illustrates how changing *graph*text attributes changes the display):

- *Rotation*. You can select the angle (0°, 90°, 180°, or 270°) of lines of text and the characters within each line of text with the `SET GRAPHTEXT ROTATION` command.
- *Path*. The `SET GRAPHTEXT CHARACTER PATH` command specifies whether characters are written above, below, to the right of, or to the left of existing characters (the directions *above*, *below*, . . . are in relation to the text orientation defined in the `SET GRAPHTEXT ORIENTATION` command — see Figure 4-13).
- *Size*. When *graph*text characters are enlarged with the `SET GRAPHTEXT SIZE` command, they appear in sizes that are integral multiples of the basic character size (5 by 7 pixels). You specify the height of *graph*text characters with the `SET GRAPHTEXT SIZE` command; the character width and intercharacter spacing are scaled so that they are proportional to the height.

Characteristic	Command	Examples ^a			
Rotation Path = RIGHT	<code>SET GRAPHTEXT ROTATION</code>				
Character Path Rotation = 0°	<code>SET CHARACTER PATH</code>				
Character Size	<code>SET GRAPHTEXT SIZE</code>	Height { Width			

^a x indicates current graphics position.

4526-8C

Figure 4-13. *Graph*text Characteristics

Using Alphatext in Graphics

As mentioned before, graphtext is better than alphatext for labels in drawings. Although you can use alphatext in the graphics area, you can't rotate or slant it.

However, you can use alphatext in drawings, provided you disable the dialog area. This feature is provided for compatibility with Tektronix 4010 Series terminals, which don't have graphtext. So, if you're emulating 4010 Series terminals, go ahead and use alphatext. Remember to put the terminal in Alpha mode before sending characters to be displayed (use the ENTER ALPHA MODE command).

You might want to set some attributes for alphatext displayed in the graphics area. The attributes you can set are the character color, the background color, and the *graphics area writing mode*. The graphics area writing mode determines whether characters overstrike or replace existing characters. You can set these attributes with the commands: SET TEXT INDEX, SET VIEW ATTRIBUTES (which controls the background color), and SET GRAPHICS AREA WRITING MODE.

GRAPHICS AREA TEXT COMMANDS

Table 4-5 summarizes the commands that affect text in the graphics area.

Table 4-5
GRAPHICS AREA TEXT COMMANDS

Command Name	Setup Name	Function
ENTER ALPHA MODE	(none)	Puts the terminal in Alpha mode
GRAPHIC TEXT	GTEXT	Writes a string of graphtext, starting at the graphics position
SET ALPHATEXT FONT	(none)	Selects the font to be used for alphatext and graphtext
SET GRAPHICS AREA WRITING MODE	GAMODE	Specifies whether the terminal overwrites or replaces characters and markers in the graphics area
SET GRAPHTEXT CHARACTER PATH	GTPATH	Specifies whether each graphtext character is displayed to the right of, to the left of, above, or below the preceding character
SET GRAPHTEXT ROTATION	GTROTATION	Specifies the angle (0°, 90°, 180°, or 270°) at which lines of graphtext are displayed
SET GRAPHTEXT SIZE	GTSIZE	Specifies the height of graphtext characters (width and spacing are scaled proportionally to height)
SET TEXT INDEX	GTINDEX	Specifies the color index for all text displayed in the graphics area

PANELS

A *panel* is a polygon that is defined by one or more more closed boundaries. A *panel definition* includes at least one `BEGIN PANEL BOUNDARY` command and terminates with an `END PANEL` command.

The area inside the panel may be empty or it may be filled with a solid color or a pattern. An area is defined as *inside* a panel if, starting from a point distant from the panel, you cross an odd number of panel boundaries to get to the area. If you cross an even number of panel boundaries or no panel boundaries, the area is *outside* the panel. See Figure 4-14 for examples of filled panels.

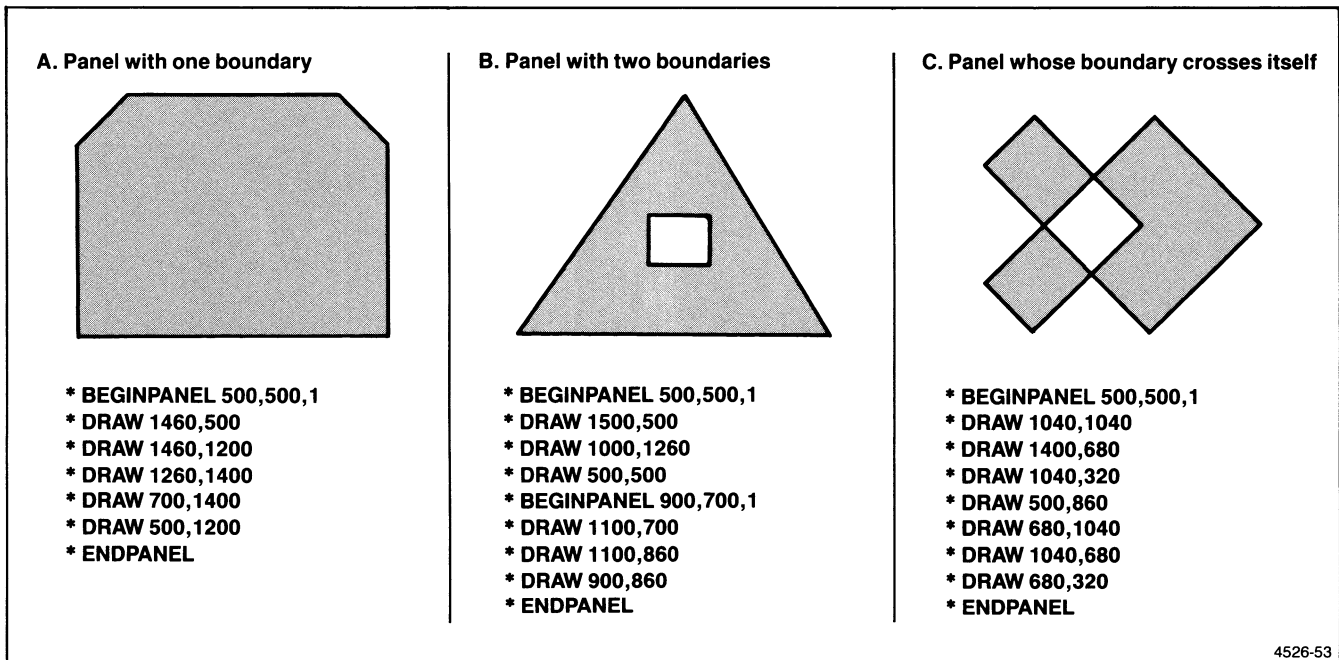
Here's how to create a panel:

1. Send the `SELECT FILL PATTERN` command and specify the fill pattern you want to use. (Appendix F shows the predefined fill patterns; also see the detailed description of this command in Section 5.)
2. Send the `BEGIN PANEL BOUNDARY` command. Use this command to specify where the panel's boundary should start and whether that boundary should be displayed in the finished panel.

3. Define the panel's boundary in either of two ways:
 - Put the terminal in Vector or Marker mode and send the endpoint coordinates of each section of the boundary (Inside a panel definition, xy-coordinates are always treated as boundary endpoints, so Marker mode works like Vector mode and doesn't display any markers).
 - Send `MOVE`, `DRAW`, or `DRAW MARKER` commands (See the discussion *Implicit Command Modes* earlier in this section).

You don't need to define the last line that closes the panel.
4. Send the `END PANEL` command. This automatically draws a line back to the starting position and fills the panel with the pattern you specified in the `SELECT FILL PATTERN` command. The graphics position is now at the panel's starting position.

Figure 4-14 illustrates several panels, each filled with a fill pattern.



4526-53

Figure 4-14. Examples of Panels.

About Multiple Panel Boundaries

As shown in Figure 4-14b, a panel can have more than one boundary. To create such a panel, issue additional BEGIN PANEL BOUNDARY commands before ending the panel. When you issue an additional BEGIN PANEL BOUNDARY command, the boundary being created is closed and a new boundary is started at the specified position. The panel is not filled until the terminal receives the END PANEL command.

PANEL COMMANDS

Table 4-6 summarizes the commands for defining panels. Turn to Appendix F to see the predefined fill patterns and colors.

Table 4-6
PANEL COMMANDS

Command Name	Setup Name	Function
BEGIN PANEL BOUNDARY	BEGINPANEL	Starts the definition of a panel boundary
END PANEL	ENDPANEL	Ends a panel definition, drawing a line to close the polygon (if necessary)
SELECT FILL PATTERN	FILLPATTERN	Specifies a color or predefined fill pattern to fill a panel

CREATING IMAGES WITH PIXEL OPERATIONS

Pixel operations offer an alternative way to create graphics images: instead of creating images from graphics primitives as described in the preceding discussion, you can assign colors to individual pixels in the display. With photographic and image processing applications, such as satellite weather maps, using pixel operations is the fastest way to display or modify images on the screen.

Your best choice between graphics primitives and pixel operations as a means to create graphics images depends on the way the image is created. Photographic images are obvious candidates for creation through pixel operations. Geometric images can be more readily created with graphics primitives.

If you don't plan to use the terminal's pixel-writing features, skip this discussion and move on to the next topic, *Using 4010 GIN (Graphics Input)*.

Pixel images are stored only in graphics memory; they cannot be stored in the terminal's program memory. Consequently, once you erase the screen, the pixel image is gone; you must retransmit it to display it again.

4105 PIXEL DIMENSIONS

Be aware that the results of pixel commands differ significantly between different Tektronix terminals, primarily because of the different dimensions of graphics memory. If you write a program that uses pixel commands for the 4105 terminal, you may have to rewrite the program to make it work with other Tektronix terminals. This is especially true if your program uses off-screen memory.

WRITING INTO THE PIXEL VIEWPORT

Pixel commands allow you to access the individual pixels in the terminal's graphics memory. Three of the commands prepare the terminal for writing pixels:

- BEGIN PIXEL OPERATIONS
- SET PIXEL VIEWPORT
- SET PIXEL BEAM POSITION

The rest of the pixel-writing commands provide varied methods of setting color indices for pixels:

- RASTER WRITE
- RUNLENGTH WRITE
- PIXEL COPY
- RECTANGLE FILL

The BEGIN PIXEL OPERATIONS Command

The BEGIN PIXEL OPERATIONS command makes two settings in preparation for pixel-writing operations.

ALU mode. The ALU mode (arithmetic logic unit) determines how the new color index information in a pixel-writing command will affect the pixel information currently stored in graphics memory — a pixel-writing operation may set pixels to a particular index, replace one image with another, or combine images. Possible values for ALU mode are:

- *A XOR B:* Combines pixel values for A and B by logically XORing the values. In this ALU mode, you can erase a pixel image by reissuing the commands that created it.
- *B:* Replaces existing image A.
- *A AND B:* Combines pixel values for A and B by logically ANDing the values.
- *A OR B:* Combines pixel values for A and B by logically ORing the values.

Bits per pixel. The number of bits-per-pixel is a value that is important to the bit-packing schemes used in the RASTER WRITE and RUNLENGTH WRITE commands. The discussions of those commands and algorithms for encoding them are just ahead.

The SET PIXEL VIEWPORT Command

The RASTER WRITE and RUNLENGTH WRITE commands operate within a *pixel viewport*, which is a rectangular area of the screen addressed in pixels. Issuing the SET PIXEL VIEWPORT command defines the size and location of the viewport.

The SET PIXEL BEAM POSITION Command

The *pixel beam position* is the position in the pixel viewport at which the RASTER WRITE and RUNLENGTH WRITE operations will begin. You control the placement of the pixel beam with the SET PIXEL BEAM POSITION command.

The coordinates of the pixel beam position are relative to the pixel viewport, not the overall graphics memory space. The coordinates 0,0 will address a different pixel if the lower-left corner of the pixel viewport is moved.

The RASTER WRITE Command

When you want to assign colors to pixels directly, you can use the RASTER WRITE command. This command takes two parameters: the number of pixels that you are encoding and a string of ASCII characters into which you have encoded the index values of the pixels.

When the terminal executes the RASTER WRITE command, it begins at the pixel beam position, fills that pixel, advances one pixel to the right, fills that pixel, and repeats until it reaches the end of the string of indices. When the terminal reaches the right edge of the pixel viewport, it moves one pixel down and wraps to the left edge of the pixel viewport. If the terminal reaches the bottom of the pixel viewport, it wraps back around to the top.

If the special character `'` (ADE 96 — left single quote) is in the ASCII string, the terminal fills the rest of the current pixel line with Index 0. The terminal then wraps back around just as if it had normally filled that line.

Encoding a RASTER WRITE Command. The encoding for the RASTER WRITE character array is called *bit packing*. The algorithm at the bottom of the page shows how to send a complete RASTER WRITE command. You can use this algorithm with any positive number of bits per pixel; however, you cannot use `'` (ADE 96) in the algorithm.

```
Procedure send-raster-write: (number-of-pixels),(index-array)
  global-reference: (bits-per-pixel)
  send-character: (ESC)
  send-character: (R)
  send-character: (P)
  send-packed-integer: (number-of-pixels)
  (number-of-characters) = integer of ((number-of-pixels)*(bits-per-pixel) + 5)/6
  send-packed-integer: (number-of-characters)
  (index-pointer) = 0
  (register) = 0
  (bits-in-register) = 0
  while (index-pointer)<(number-of-pixels):
    increment (index-pointer)
    (index) = (index-array(index-pointer))
    shift (register) left (bits-per-pixel)
    increment (register) by (index) modulo 2**((bits-per-pixel))
    increment (bits-in-register) by (bits-per-pixel)
    while (bits-in-register) = > 6:
      (char-to-send) = (shift (register) right ((bits-in-register)-6) + 32)
      send-character: (char-to-send)
      /* Clear leftmost 6 bits from register as follows */
      and (register) with (2**((bits-in-register)-6)-1)
      decrement (bits-in-register) by 6
  if (bits-in-register) > 0:
    shift (register) left 6-(bits-in-register)
    send-character: (register) + 32
```

The RUNLENGTH WRITE Command

The RUNLENGTH WRITE command is similar to the RASTER WRITE command — use it when you want to set most of the pixels on a line to the same index. The RUNLENGTH WRITE command requires less characters for specifying pixel data than does the RASTER WRITE command.

You must specify colors in an array of *runcodes*. Each runcode is a single number into which two other numbers are packed. The runcodes are packed using this form:

$$\text{Runcode} = \text{number-of-pixels} \times 2^n + \text{color-index}$$

The *bits-per-pixel* parameter from the most recent BEGIN PIXEL OPERATIONS command supplies the value for *n*, unless that parameter is 4 or 6; then the value of *n* is 3.

Encoding a RUNLENGTH WRITE Command. The algorithm at the bottom of the page shows how to encode the RUNLENGTH WRITE runcodes.

The RECTANGLE FILL Command

If you want to fill a rectangular area of the screen with a single color index, you can use the RECTANGLE FILL command. This command does not need to be within the pixel viewport.

The PIXEL COPY Command

The PIXEL COPY command copies pixels from one rectangular region in graphics memory to another rectangular region of graphics memory. Refer to the command descriptions in the Section 5 for more information about the PIXEL COPY commands.

```

Procedure send-runlength-write: (number-of-pixels),(index-array)
  global-reference: (bits-per-pixel),(terminal-model)
  local-array: (code-array)
  send-character: (ESC)
  send-character: (R)
  send-character: (L)
  (code-count) = 0
  (index-pointer) = 1
  (multiplier) = 2**(bits-per-pixel)
  (index) = (index-array(1))
  (index-count) = 1
  (max-index-count) = integer of 65535/(multiplier)
  until (index-pointer) = (number-of-pixels)
    increment (index-pointer)
    if (index) > (index-array(index-pointer))
      or (index-count) = (max-index-count)
      increment (code-count)
      if (index) ≥ (multiplier)
        (index) = (multiplier)-1
        (code-array(code-count)) = (multiplier)*(index-count) + (index)
        (index) = (index-array(index-pointer))
        (index-count) = 1
      else
        increment (index-count)
  send-packed-integer: (code-count)
  for (counter) = 1 to (code-count)
    send-packed-integer: (code-array(counter))

```

USING PIXEL OPERATIONS: AN EXAMPLE

Figure 4-15 shows the commands that define a pixel viewport and write color indices into that viewport. Let's consider these commands, one by one.

PXBEGIN. The **PXBEGIN** (**BEGIN PIXEL OPERATIONS**) command must be issued before any other pixel-writing commands. In this example, no parameter values are given, so the terminal will assume the command's default parameter values 11 and 6. These parameters set the ALU mode to 11 (Replace mode) and specify that six bits per pixel be used to transmit each color index (the 4105 will use three bits per pixel, since that is its maximum).

PXVIEWPORT 100,100,109,109. The **PXVIEWPORT** (**SET PIXEL VIEWPORT**) command defines a rectangular region on the pixel-writing surface. In this example, the lower-left and upper-right corners of the pixel viewport are at 100,100 and 109,109, respectively. These coordinates are in 480x360 graphics memory space.

PXRECTANGLE 100,100,109,109,0. The **PXRECTANGLE** (**RECTANGLE FILL**) command sets all pixels within a rectangular region to the same color index. In this example, the command clears the pixel viewport by setting all pixels in that region to Index 0.

PXPOSITION 3,4. The **PXPOSITION** (**SET PIXEL BEAM POSITION**) command moves the current pixel position (the point where the next pixel will be written) to coordinate point 3,4. These coordinates are relative to the lower-left corner of the pixel viewport.

PXRASTERWRITE 4,"4002". The **PXRASTERWRITE** (**RASTER WRITE**) command writes the Indices 4, 0, 0, and 2 into four successive pixels. At the end of this command, the pixel beam position is at 7,4. These coordinates are relative to the lower-left corner of the pixel viewport.

PXPOSITION 3,3. This command moves the current pixel position to the location 3,3 in pixel viewport coordinates.

PXRASTERWRITE 4,"4217". This command writes the Indices 4, 2, 1, and 7 into four successive pixels. At the end of this operation, the pixel beam position is at 7,3 in pixel viewport coordinates.

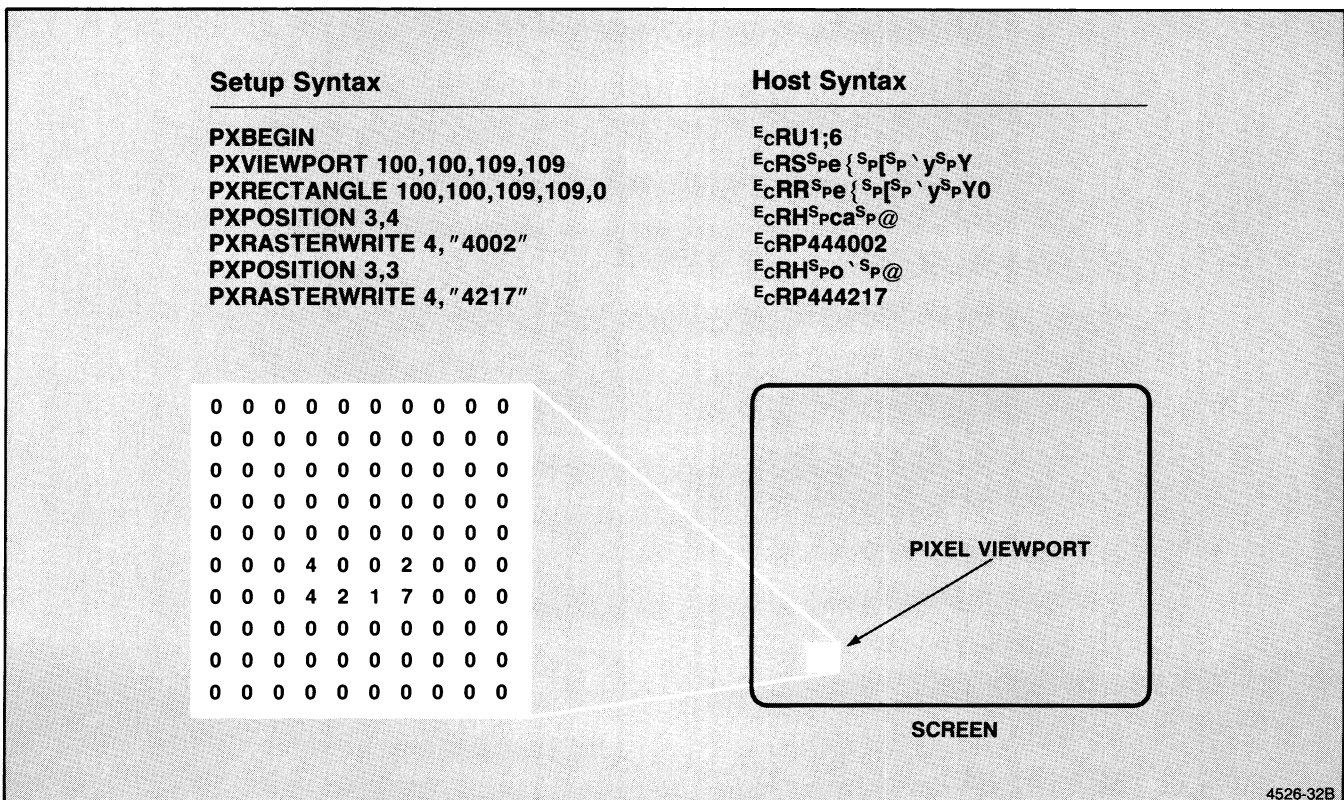


Figure 4-15. Writing Into the Pixel Viewport Using RASTER WRITE.

Recall that you can use RUNLENGTH WRITE commands as well as RASTER WRITE commands to write in the pixel viewport. RUNLENGTH WRITE (using runcodes) specifies the number of pixels in a sequence, and sends the same color index to each pixel in the sequence. The next three paragraphs and Figure 4-16 explain how this is done.

As in Figure 4-15, Figure 4-16 uses PXBEGIN (this time with bits-per-pixel set to 3) and PXVIEWPORT commands to define a pixel viewport that is 10 pixels wide and 10 pixels high. As before, a PXRECTANGLE command clears all pixels in the pixel viewport to Index 0. This time, however, there is no PXPOSITION command, so the current pixel position starts at the upper-left corner of the pixel viewport.

In this example (Figure 4-16), the RUNLENGTH WRITE command has four runcodes in its integer-array parameter.

The first code, 160, calls for 20 pixels displayed in Index 0 ($20 \times 2^3 + 0 = 160$). The second code, 243, calls for 30 pixels displayed Index 3 ($30 \times 2^3 + 3 = 243$). The third code, 160, calls for 20 pixels displayed in Index 0; it is the same as the first code. The fourth code, 246, calls for 30 pixels of Index 6 ($30 \times 2^3 + 6 = 246$.)

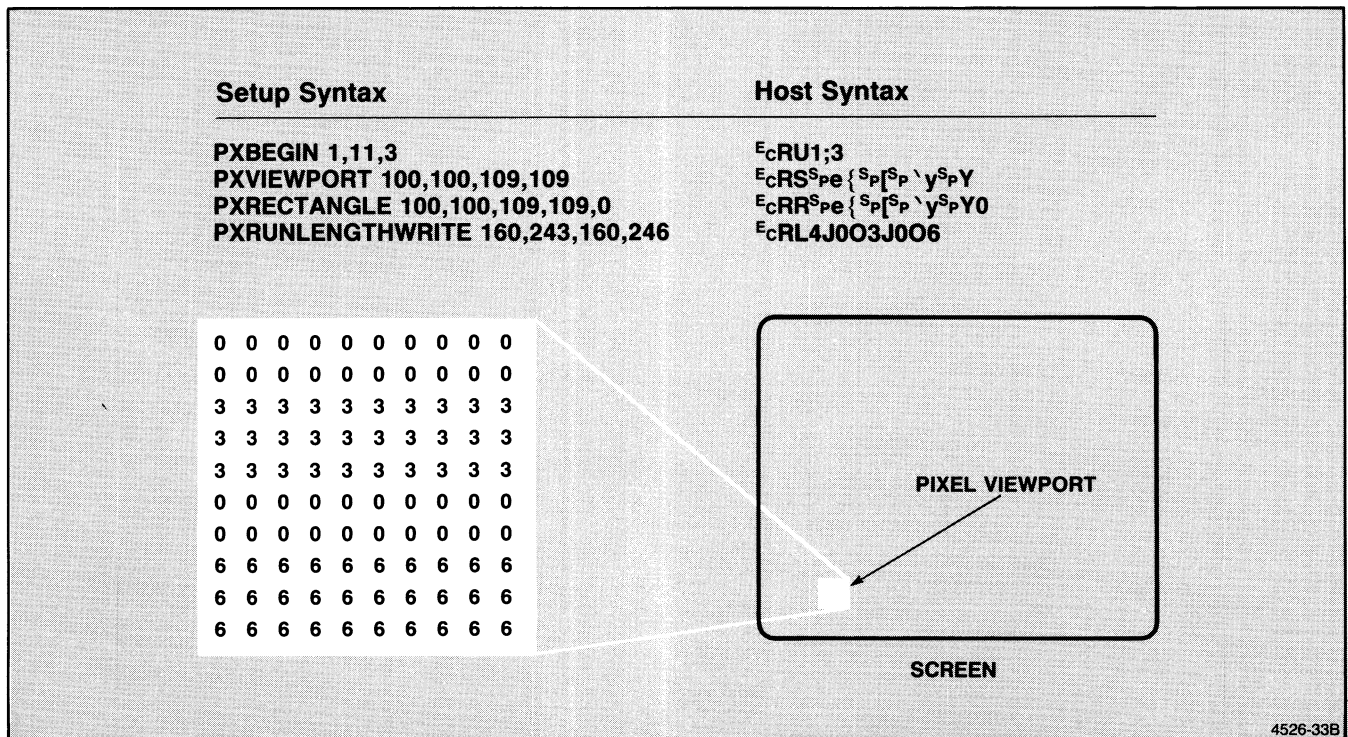


Figure 4-16. Writing Into the Pixel Viewport Using RUNLENGTH WRITE.

PIXEL COMMANDS

The commands that you use in performing pixel operations are listed in Table 4-7.

Table 4-7
PIXEL COMMANDS

Command Name	Setup Name	Function
BEGIN PIXEL OPERATIONS	PXBEGIN	Sets up the terminal for pixel operations
PIXEL COPY	PXCOPY	Copies pixels from one rectangular region to another
RASTER WRITE	PXRASTERWRITE	Sets the color indices individually for one or more pixels in the pixel viewport
RECTANGLE FILL	PXRECTANGLE	Sets all the pixels in a rectangle to the same color
RUNLENGTH WRITE	PXRUNLENGTHWRITE	Sets one or more pixels in the pixel viewport to the same color
SET PIXEL BEAM POSITION	PXPOSITION	Sets the position of the pixel beam in the pixel viewport
SET PIXEL VIEWPORT	PXVIEWPORT	Sets the pixel viewport's size and position in graphics memory

USING 4010 GIN

In many graphics applications, users must choose a menu option from the terminal screen or designate a location on the screen for graphics or text, and the terminal must transmit those choices to the host program in terms of xy-coordinates. *GIN* (or, *Graphic Input*) allows users to indicate their choices to the host without calculating the xy-coordinates. The 4105 terminal supports 4010-style GIN, which makes the terminals compatible with programs written for Tektronix 4010 Series terminals.

Programming 4010 GIN is straightforward; it works like this:

1. Your program issues the ENABLE 4010 GIN command, which enables the terminal for one 4010 GIN Report and causes a crosshair cursor to be displayed on the terminal screen.
2. The user presses the Joydisk to move the cursor to a location on the screen. The user can press the Shift key to slow the cursor's movement for more accurate control (the SET GIN CURSOR SPEED command controls the normal and shifted speeds).
3. The user *inputs* the selected location by pressing any keyboard key that generates an ASCII character; this causes the terminal to generate a report to the host.
4. The terminal sends your program a 4010 GIN Report which indicates the location of the GIN cursor and the key the user pressed.
5. Your program must parse the GIN report.
6. After sending the 4010 GIN Report, the terminal updates its graphics position to the GIN cursor location, and enters Alpha mode.

Since the terminal is enabled for only one report, a host program must issue a separate command for each GIN point required.

If, instead of pressing a single key, the user presses several keys in succession or presses a key that has a key macro defined for it, the terminal still sends a 4010 GIN Report, but the excess characters are transmitted to the host. The first key pressed or the first character of the macro string becomes the key-character sent in the 4010 GIN Report, and the rest of the characters are transmitted to the host.

Communication Settings for 4010 GIN

Emulating the graphics input capability of a Tektronix 4010 Series terminal requires some specific communications settings. While the exact settings may vary from one computer installation to another, the settings shown for the following commands should work for most host computers:

- SET EOM CHARACTERS — Set both EOM characters to `NU`.
- SET EOL STRING — Set the EOL string to `CR` (this is its default value).
- SET BYPASS CANCEL CHARACTER — Set the bypass cancel character to match to the last character that the host echoes in response to a report's last character. For 4010 GIN, the bypass cancel character will be one of these:
 - `NU` if the host is not echoing characters
 - `LF` if the host echoes `CR` as `CRLF`
 - `CR` if the host echoes `CR` as just `CR`

USING MACROS

You can store strings of characters — either commands or text — in the terminal's program memory so that they can be accessed by a single command or keystroke. These strings are called *macros*, and each macro is identified by an integer, called the *macro number*.

Assigning a frequently used command sequence to a single key can make a program easier and more convenient to use — you can see this illustrated in the examples included in this discussion. You can also program keys right from the keyboard, custom tailoring the terminal to meet your individual needs. For instance, you could program a key so that it moves the alpha cursor to the home position.

Using macros reduces data transmission time. Instead of sending a long string of characters, the host can (1) send a single command, `EXPAND MACRO`, which calls the stored string from program memory, or (2) program a key so that just a single keystroke executes or displays the string.

VOLATILE AND NONVOLATILE MACROS

The terminal can store a macro temporarily or permanently. A *volatile macro* is stored in program memory — the macro is not retained when the terminal is turned off or reset. A *nonvolatile macro* is stored in nonvolatile memory — the macro is retained even when the terminal is turned off or reset.

You can store a macro in both volatile memory and nonvolatile memory at the same time. When expanding a macro, the terminal checks the volatile memory first. If it does not find the macro in volatile memory, the terminal looks for it in nonvolatile memory.

The `DEFINE MACRO` and `LEARN` commands define only the volatile version of a macro. The `DEFINE NONVOLATILE MACRO` and `LEARN NONVOLATILE` commands define both the volatile and nonvolatile versions of a macro. To actually save the nonvolatile version in nonvolatile memory, however, you must issue a `SAVE NONVOLATILE PARAMETERS` command before turning off or resetting the terminal.

The `DEFINE MACRO` and `DEFINE NONVOLATILE MACRO` commands work in the same way — one is for *volatile* (not permanent) macros and the other is for *nonvolatile* (permanent) macros. In this discussion, wherever you read `DEFINE MACRO`, you could substitute `DEFINE NONVOLATILE MACRO` — the information applies to both. The same goes for `LEARN` and `LEARN NONVOLATILE`, which are discussed under *Defining Key Macros*.

MEMORY REQUIREMENTS FOR MACROS

To define a macro with the `DEFINE MACRO` or `DEFINE NONVOLATILE MACRO` command, you need program memory both for processing the commands and for storing the volatile version of the macro. The `DEFINE NONVOLATILE MACRO` also requires nonvolatile memory for storing the nonvolatile version of the macro.

Each volatile or nonvolatile macro you define takes eight bytes of volatile memory for header information, plus one byte for each character in the definition. (Nonvolatile macros take the same amount of nonvolatile memory besides.) The maximum size of a macro depends on the availability of program memory and on which command you use to define it:

- The `DEFINE MACRO` command issued from the host lets you use all the terminal's available volatile memory for a macro definition.
- The `DEFINE MACRO` command issued from Setup limits you to a single line of entry — the macro contents can be up to 110 characters.
- The `LEARN` command issued from Setup lets you use up to one-half the terminal's available volatile memory for a macro definition.
- The `DEFINE NONVOLATILE MACRO` command and the `LEARN NONVOLATILE` command limit macro size depending on the availability of both volatile memory and nonvolatile memory. When no macros have been defined, there is enough nonvolatile memory for about 1500 characters of macro definition.

You can use the `REPORT TERMINAL SETTINGS` command (or issue `STATUS MEMORYBLOCKS` from Setup) to find out how much volatile memory is available. You can use all the available volatile memory for one macro definition, or you can define many shorter macros.

If there is not enough volatile memory to define a macro (or not enough nonvolatile memory to store the nonvolatile version of a macro), the terminal issues an error message. To define a macro when you've run out of memory, you must free some memory:

- To free volatile memory, you must delete macros or reduce dialog buffer size or queue size. See *Managing Program Memory* later in this section for an explanation of how the dialog buffer size and queue size affect volatile memory.
- To free nonvolatile memory, you must delete nonvolatile macros.

DEFINING MACROS

From the Host. Use the DEFINE MACRO or DEFINE NONVOLATILE MACRO command to define macros from the host. In the command, you give the number that you are assigning to the macro and the ASCII decimal equivalents (ADE) of the characters that make up the macro's content. You must convert the ADE values of characters to encoded integers, as explained in the first part of Section 5; for most integers, you can simply refer to Appendix D, which is a list of integers already converted to their encoded version.

An example showing how to define a macro from the host is included at the end of this discussion.

From the Keyboard. You can use either the Setup version of the DEFINE MACRO command or the LEARN command to define macros from the keyboard. (To define nonvolatile macros, use the DEFINE NONVOLATILE MACRO or LEARN NONVOLATILE command.) The following example shows how to use the DEFINE NONVOLATILE MACRO command.

An example showing how to define a macro from the keyboard is included at the end of this discussion.

EXECUTING MACROS

From the Host. Once you've defined a macro, you can invoke it from the host by issuing the EXPAND MACRO command. The terminal calls the macro from program memory and *expands* it — that is, displays the macro's contents if the characters are a string of text, or executes the macro if the macro's contents make up a command sequence.

For example, a host program can define Macro 301 to be the string, "Type any key to continue." Then, in your program, anytime you need this string displayed, just send an EXPAND MACRO command for Macro 301. When the terminal receives this command, it looks up Macro 301 in program memory and displays the string.

Likewise, you could define Macro 302 to execute a frequently used sequence of commands — perhaps a sequence that resets the dialog area to a known state. Thereafter, you could issue the single command EXPAND MACRO for Macro 302 — rather than repeatedly reissuing the lengthy and time-consuming sequence — to put the terminal in that state. As you can see, macros help you use the terminal's features efficiently.

From the Keyboard. To expand a macro from the keyboard, users can enter Setup and issue the EXPAND MACRO command in Setup syntax with the macro number. The terminal routes the characters in the macro definition through the terminal's Setup command interpreter. The terminal executes any commands in the macro and sends all other alphanumeric data to the dialog area.

DEFINING KEY MACROS

If you define a macro using a macro number between -150 and 143, users can expand the macro just by pressing a keyboard key or key combination. This is possible because each number in that range (except -1, which is used for deleting macros) has a unique association with a key or key combination. For instance, if you define Macro -42, you can expand it by holding down the Ctrl and Shift keys while pressing the Back Space key (Ctrl-Shift-BackSpace).

As you might expect, macros associated with keys in this way are called *key macros*. Appendix A shows the numbers assigned to keys.

Executing Key Macros

To expand a key macro, the user simply types the programmed key or key combination. The terminal sends the information in the macro to the host — or uses the information locally if the definition includes key-execute characters. Key-execute characters are explained just ahead under *Keeping a Key Macro Local*.

Disabling Key Macros

You can disable key macros to prevent users from expanding a key macro at an inopportune time. If your program expects a key's default meaning, but you've assigned macros to many of the keyboard keys, it would be wise to disable key macros.

Use the `ENABLE KEY EXPANSION` command to disable or enable key macros. When key expansion is disabled, users can't expand key macros with a keystroke, but you (or the user) can still expand them by issuing the `EXPAND MACRO` command.

Key macros are automatically disabled whenever a `SELECT CODE` command puts the terminal in `EDIT` mode.

NOTE

Although key macros are automatically disabled when you enter `EDIT` mode from `TEK` mode, they are not automatically reenabled when you reenter `TEK` mode — you must specifically reenable them with the `ENABLE KEY EXPANSION` command either from the host or from the keyboard.

Keeping a Key Macro Local

Normally, when you press a key that has a macro definition, the terminal sends the macro to the host — just as if you had typed that sequence of characters on the keyboard. Sometimes, however, you might want a key macro to be processed by the terminal rather than transmitted to the host. For instance, if the macro contains text for display on the screen or a command to change some terminal setting, you'd want the macro executed locally, not sent to the host.

When the terminal finds a *key-execute character* in a macro, it executes the subsequent characters locally — until it comes across another key-execute character. The `␣` character is the terminal's default key-execute character, but you can change it with the `SET KEY EXECUTE CHARACTER` command.

To create a locally executed macro, send the `DEFINE MACRO` command and start the definition with the key-execute character. Following the string of characters to be used locally, enter another key-execute character. When a user presses the defined key, the terminal interprets the string of characters between the two key-execute characters locally and does not send them to the host.

If the string is a terminal command, the terminal executes it. For example, you could define Function Key F4 to set the dialog area to eight lines. Use the `LEARN` command to program the key from the keyboard. The terminal will prompt you for the key to be defined (just press the F4 key) and for the definition. When it prompts you for the definition, you would enter:

`Ctrl-P Esc LL8 Ctrl-P`

(`Ctrl-P` is the key combination that you enter from the keyboard to transmit `␣`, the default key-execute character.)

Now when you press the F4 key, the terminal will assign eight lines to the dialog area. However, since the definition uses Host syntax, the terminal cannot be in Setup (pressing F4 in Setup will just display the string `␣ LL8`).

If the string between the two key-execute characters is not a terminal command, the terminal displays the characters in the string as if the host had sent them. For example, if you define F5 with the string `Ctrl-P Hello Ctrl-P` and then press F5 (the terminal must not be in Setup), the terminal displays "Hello" in the dialog area, but the characters are not sent to the host.

CAUTION

When defining a macro to be used locally, be sure to include the second key-execute character. Otherwise, the terminal will continue to interpret characters locally until it encounters another key-execute character.

Key-execute characters in macros expanded by an `EXPAND MACRO` command have no special meaning; they are treated just like other characters in the macro.

Another way to keep a key macro local is to define the macro containing the Setup version of a command instead of the host version. If you use the `LEARN` command to do this, you can simply enter the definition — you can even include a Carriage Return at the end of the definition to terminate the command. If you use the `DEFINE MACRO` command to define the macro, however, you will need to precede the Carriage Return with the *literal edit character*, which allows you to include Carriage Return or other edit characters in a macro definition (see the `SET EDIT CHARACTERS` command).

To use the macro, enter Setup, and press the defined key. If you included a Carriage Return in the definition, the terminal will execute the command. If you didn't include a Carriage Return, the terminal will display the command but won't execute it until you press the Return key.

DEFINING MACROS FROM THE HOST: AN EXAMPLE

The Scenario: Suppose that your application uses the bottom six lines of the screen to display text, and you want a frame around the part of the screen used to display graphics. Rather than issuing the same commands each time you want to redisplay the frame, you can define a macro that contains the commands — and then expand the macro each time you want to redraw the frame.

Figure 4-17 shows the commands that you would issue from the host to draw the frame, and it also shows the steps that you would take to encode these commands so you can include them in the DEFINE MACRO command. The commands shown program Macro 200 to draw a dotted-line frame around the top three-fourths of the screen.

You must encode the characters that make up the commands that you include within the macro definition.

Here's what the command itself would look like (spaces are included in the command for readability; they must *not* be included when your program issues this command):

```
EcKD L8 B2 A0 A;D = E61 A = B6F0F0B0D0 B6F3F0C?E?
C8F3F?C?E? C8F0F?B0D0 B6F0F0B0D0 A0
```

Here's a breakdown of the command:

EcKD

This is the escape code for the DEFINE MACRO command.

L8

This is the encoded value of 200 (the macro number).

B2

This is the encoded value of 34, the array count (there are 34 encoded integers following).

A0

This is the encoded value of 16, the ADE of **D** (the first key-execute character); it tells the terminal to process the following characters.

...

(The description of the encoded form of each xy-coordinate is included in Figure 4-17.)

A0

This is the encoded value of 16, the ADE of **D** (the second key execute character); it returns control to the host application.

Now you can just expand Macro 200 each time you want to draw the frame — just issue the EXPAND MACRO command, **EcKXL8**.

Command	Host Syntax ^a	ADE	Encoded for Macro Definition
LINE STYLE 1	EcMV1	27 77 86 1	A;D = E61
ENTER VECTOR MODE	G _S	29	A =
MOVE 0,768	& \ 1 Sp@	38 96 96 32 64	B6F0F0B0D0
DRAW 4095,768	&c \ ? _	38 99 96 63 95	B6F3F0C?E?
DRAW 4095,3132	8co? _	56 99 111 63 95	C8F3F?C?E?
DRAW 0,3132	8 \ o Sp@	56 96 111 32 64	C8F0F?B0D0
DRAW 0,768	& \ 1 Sp@	38 96 96 32 64	B6F0F0B0D0

^a The MOVEs and DRAWs issued in host syntax are implicit commands. Since the first command is ENTER VECTOR MODE, these commands omit the host escape sequence and consist of xy-coordinates only.

4526-55

Figure 4-17. Encoding a Macro.

DEFINING MACROS FROM THE KEYBOARD: AN EXAMPLE

The Scenario: Suppose you are writing an application for users familiar with DEC VT100 terminals, and you want to emulate the VT100 NO SCROLL key, which stops and starts host communication.

You can program any key to use the D_1 and D_3 characters to emulate the function of the NO SCROLL key. Entering a D_3 character at your keyboard stops transmission from the host (and thus stops scrolling), and entering a D_1 character starts it up again.

This example shows macro definitions that program the Vertical Bar key (|) so that it:

1. Sends a D_3 to the host, thus stopping transmission from the host
2. Reprograms the Vertical Bar so that the next time you press it, it will send a D_1 to the host and restart data transmission
3. Reprograms the Vertical Bar back to its original definition — so that the next time you press it, it will again send D_3

The keystroke that resumes scrolling must put the terminal back in ANSI mode.

NOTE

In Setup, you need to press key combinations to generate the D_1 , D_1 , and D_3 characters at the keyboard. When you enter Ctrl-P, the terminal displays D_1 . When you enter Ctrl-Q or Ctrl-S, the terminal displays D_1 or D_3 , respectively. (Remember, the key combinations Ctrl-P and Ctrl-S mean “press the Ctrl key and the P or S key simultaneously,” and E_c means the Esc key.)

This example contains three levels of commands. You issue NVDEFINE Setup commands to define Macros 124, 145, and 146; these macros contain host-syntax commands that issue other commands. Each level of command requires its own level of encoding:

1. When you issue the NVDEFINE commands, you use Setup syntax. You type the macro number and the delimiters for the macro string; use key combinations to generate control characters D_L , D_1 , and D_3 ; and type the host command.
2. When you define commands within a macro you're creating with NVDEFINE, you use host syntax because the terminal will no longer be in Setup when the macros are expanded. You press the Esc key for E_c , type the opcode, and encode parameter values for TEK mode commands. Macros 124, 145 and 146 use the TEK mode commands DEFINE MACRO and EXPAND MACRO.
3. The host syntax of the TEK mode commands (DEFINE MACRO and EXPAND MACRO) require encoded integer arrays. The commands defined within the DEFINE MACRO and EXPAND MACRO commands also use host syntax, but the entire command string must be encoded as an integer array. You must convert each character to its ADE value and then encode each ADE value as an integer parameter.

To program the Vertical Bar key, place the terminal in Setup and type:

```
KEYEXPAND YEScr
NVDEFINE 145,/ Ctrl-P EcKDG<<A1A0A;D;E8D9C2A;B5B1C1A0 Ctrl-P /cr
NVDEFINE 146,/ Ctrl-P EcKDG<<A3A0A;B5B1C0A;D;E8D9C1A0 Ctrl-P /cr
NVDEFINE 124,/ Ctrl-S Ctrl-P Ec%!0EcKXI1 Ctrl-P /cr
NVSAVEcr
```

The first definition creates Macro 145, which reprograms the Vertical Bar key so that the next time you press it, it will send a ^{D₁} to the host and expand Macro 146.

The entries have the following meanings:

NVDEFINE

This is the Setup name of the DEFINE NONVOLATILE MACRO command.

145

This is the macro number.

/

This slash marks the beginning of the string to be defined as a macro.

Ctrl-P

This is the key combination you use to type the ^{D_L} character, which is the terminal's default key-execute character. The key-execute character starts macro expansion at the terminal. (In this case what is expanded is another macro definition.)

^{E_c}**KD**

This is the host escape sequence for the DEFINE MACRO command.

G<

This is the encoded value of the integer 124 — the macro number assigned to the Vertical Bar (|).

<

This is the array count. It is the encoded value of the integer 12, since there are 12 integers are in the array.

A1

This is the encoded value of 17, which is the ADE of ^{D₁}, the flagging character that starts transmission of data from the host.

A0

This is the encoded value of 16, which is the ADE of ^{D_L}, the first key-execute character, which starts expansion of the macro at the terminal.

A;D;E8D9C2

This is the encoded value of the EXPAND MACRO command ^{E_c}**KXI2**, which expands Macro 146.

Character	ADE	Encoded Value
^{E_c}	27	A;
K	75	D;
X	88	E8
I	73	D9
2	50	C2

A;B5B1C1

This is the encoded value of the SELECT CODE command, ^{E_c}%!**1**, which puts the terminal in ANSI mode.

Character	ADE	Encoded Value
^{E_c}	27	A;
%	37	B5
!	33	B1
1	49	C1

A0

This is the encoded value of 16, which is the ADE of ^{D_L}, the key-execute character. The second occurrence of ^{D_L} returns control to the host.

Ctrl-P

This is the key combination you use to type the ^{D_L} character, which is the terminal's default key-execute character. The second occurrence of the key-execute character stops macro expansion at the terminal and returns control to the host.

/

This slash marks the ending of the string to be defined as a macro.

MACROS

The second definition creates Macro 146, which reprograms the Vertical Bar back to its original definition — the next time you press it, it will again send D_3 to the host and expand Macro 145.

The entries have the following meanings:

NVDEFINE

This is the Setup name of the DEFINE NONVOLATILE MACRO command

146

This is the macro number.

/

This slash marks the beginning of the string to be defined as a macro.

Ctrl]-P

This is the key combination you use to type the D_L character, which is the terminal's default key-execute character. The key-execute character starts macro expansion at the terminal. (In this case what is expanded is another macro definition.)

E_C KD

This is the host escape sequence for the DEFINE MACRO command.

G<

This is the encoded value of the integer 124 — the macro number assigned to the Vertical Bar (|).

<

This is the array count. It is the encoded value of the integer 12, since there are 12 integers are in the array.

A3

This is the encoded value of 19, which is the ADE of D_3 , the flagging character that stops transmission of data from the host.

A0

This is the encoded value of 16, which is the ADE of D_L , the key-execute character. The first key-execute character starts expansion of the macro at the terminal.

A;B5B1C0

This is the encoded value of the SELECT CODE command, $E_C \% ! 0 <$, which puts the terminal in TEK mode.

Character	ADE	Encoded Value
E_C	27	A;
$\%$	37	B5
!	33	B1
0	48	C0

A;D;E8D9C1

This is the encoded value of the EXPAND MACRO command, $E_C K X I 1$, which expands macro 145 (encoded I).

Character	ADE	Encoded Value
E_C	27	A;
K	75	D;
X	88	E8
I	73	D9
1	49	C1

A0

This is the encoded value of 16, which is the ADE of D_L , the key-execute character. The second key-execute character returns control to the host.

Ctrl]-P

This is the key combination you use to type the D_L character, which is the terminal's default key-execute character. The second occurrence of the key-execute character stops macro expansion at the terminal and returns control to the host.

/

This slash marks the ending of the string to be defined as a macro.

The third definition creates a key macro for the vertical bar key. When you press the Vertical Bar key, the terminal sends a D_3 to stop scrolling, enters TEK mode (so that macro expansion is possible), and expands Macro 145.

The entries have the following meanings:

NVDEFINE

This is the Setup name of the DEFINE NONVOLATILE MACRO command.

124

This is the macro number.

/

This slash marks the beginning of the string to be executed.

Ctrl]-S

This is the key combination you use to type the D_3 character, which causes the terminal to stop scrolling.

Ctrl]-P

This is the key combination you use to type the D_L character, which is the terminal's default key-execute character. The first key-execute character starts macro expansion at the terminal.

$E_C \% ! 0$

This is the SELECT CODE command to put the terminal in TEK mode.

E_c KX11

This is the command to expand Macro 145.

Ctrl]-P

This is the key combination you use to type the D_L character, which is the terminal's default key-execute character. The second key-execute character returns control to the host.

If you want to program a different key as the NO SCROLL key, just make the following changes to the sequence used in this example:

1. Change the key specifier in the last definition from 124 to the macro number of the key you prefer (see the keyboard charts in Appendix A).
2. In the first and second key definitions, replace the $G<$ with the encoded integer parameter for the macro number of the key.

DEFINING KEY MACROS: AN EXAMPLE

The Scenario. Suppose that you are writing a host program and want to make it easy for the user to create a log of text written to the dialog area. You can select this sort of data logging with the ANSI MC (MEDIA COPY) command. This example shows you how to program the F8 function key to issue the MEDIA COPY command toggle data logging on and off. Then, when in ANSI mode, the user can press the F8 key to switch the data logging feature on and off.

NOTE

The macro contents in this example must be encoded because your program must issue the DEFINE MACRO command from the host with the terminal in TEK mode. Since the macro contents include the ANSI MC (MEDIA COPY) command, you must encode the characters that make up this command. Also, the example uses the default key-execute character D_L .

To program the F8 function key to switch data logging on and off, place the terminal in TEK mode and issue the following from the host:

E_c KDH77A0A;E;C?3F9A0

The entries that make up this macro definition are explained individually in the following discussion:

E_c KD

This is just the escape sequence for the DEFINE MACRO command.

H7

This is the *macro-number* parameter for the DEFINE MACRO command. It is the encoded value of the number 135, which is the macro number assigned to the F8 function key.

7

This is the array count that begins the *macro-contents* parameter of the DEFINE MACRO command. It is the number of encoded integers that follow in the array. Note that the array count itself must be an encoded integer — in host syntax, 7 is the encoded value of 7.

A0

This is the encoded value of the number 16, which is the ADE of D_L , the default key-execute character. The first occurrence of the key-execute character tells the terminal not to transmit subsequent characters to the host, but to execute them locally.

A;E;C?3F9

This is how you encode E_c {?3i, which is the host syntax for the ANSI MC (MEDIA COPY) command, with parameter value ?3 (toggle data logging):

Character	ADE	Encoded Integer
E_c	27	A;
[91	E;
?	63	C?
3	3	3
i	105	F9

A0

This is the encoded value of the number 16, which is the ADE of D_L , the default key-execute character. The second occurrence of the key-execute character tells the terminal to quit executing the transmitted characters at the terminal and to transmit subsequent characters to the host.

Now, when in ANSI mode, users can simply press the F8 Key to switch data logging on and off.

MACROS

DELETING MACROS

You can delete a macro by redefining it as an empty string. Just issue **DEFINE MACRO** and give the macro number, but don't put anything in the macro's contents (in host syntax, issue an empty array for the macro contents). For example, to delete Macro 500 from the keyboard, you would just issue:

Host syntax: `^cKD_40`
 Setup syntax: `DEFINE 500`

In the host syntax, the `_4` is the encoded integer for `500`, and the `0` says that the array is empty.

If you want to delete all macros, issue **DEFINE MACRO** with `-1` (in host syntax) or `all` (in Setup syntax) as the macro number. In other words, to delete all macros you'd issue:

Host syntax: `^cKD!0`
 Setup syntax: `DEFINE ALL`

The `!` following the escape sequence is the encoded integer for `-1`. Although you can omit the last parameter of a command in host syntax, it's safer not to — the `0` specifies that the array is empty.

MACRO COMMANDS

Table 4-8 summarizes the commands for defining and executing macros.

Table 4-8
MACRO COMMANDS

Command Name	Setup Name	Function
DEFINE MACRO	DEFINE	Creates or deletes volatile macros
DEFINE NONVOLATILE MACRO	NVDEFINE	Creates or deletes nonvolatile macros, which can be save in nonvolatile memory
ENABLE KEY EXPANSION	KEYEXPAND	Controls whether or not key macros can be expanded by pressing keyboard keys
EXPAND MACRO	EXPAND	Expands a macro
LEARN	LEARN	Creates or deletes volatile key macros
LEARN NONVOLATILE	NVLEARN	Creates or deletes nonvolatile key macros, which can be saved in nonvolatile memory
MACRO STATUS	MACROSTATUS	Displays the definition of a macro
SAVE NONVOLATILE PARAMETERS	NVSAVE	Saves the values of those commands whose settings can be saved in nonvolatile memory; also saves macros created with the DEFINE NONVOLATILE MACRO and LEARN NONVOLATILE commands
SET EDIT CHARACTERS	EDITCHARS	Specifies the special editing characters used in the dialog area while in Setup
SET KEY EXECUTE CHARACTER	KEYEXCHAR	Specifies the character used in key macros to specify whether subsequent characters should be processed by the host or by the terminal

PUTTING TOGETHER A GRAPHICS PROGRAM

The way you use the terminal's features largely depends on the needs of your application — we won't try to outline an exact sequence of commands that will meet the needs of your application. Yet, because of the hierarchical structure of the terminal's features, we can give you some general principles that apply to writing a graphics program. For example, some commands set up conditions that affect commands issued later — a program must select the attributes for a graphics primitive before specifying the primitive; that's like deciding whether to use a pen or a pencil before drawing a line on paper.

After you have absorbed the graphics concepts explained in this section, you can take a larger view of how the terminal's features interact. We've put together a summary of the graphics concepts, and listed them from top to bottom in the order that you should follow to structure your application program. The list doesn't include all the features or subtle interactions between commands — it covers just those highlights that typically are troublesome to someone new to graphics programming. Following the general recommendations given here can help you avoid much trial and error in putting together command sequences.

When designing the graphics displayed by the program, you'll probably want to first lay out the graphics primitives needed for an image, and then figure out the larger context for displaying the image. You should issue the commands to the terminal, however, in the suggested order.

Here's the list, starting at the top with those items that a program should configure first:

- *Set up communications* — command syntax mode, special communications settings, and report characteristics
- *Set up dialog and keyboard* — dialog attributes for messages to users, and keyboard macro definitions; modify later as needed
- *Manipulate graphics images* — change attributes of graphics images according to user input
- *Restore parameters* — at end of application program, restore terminal's parameters to initial state, allowing a smooth transition to next program that uses terminal

To study the details of command interactions, refer to the functionally grouped list of commands on the tab labeled *Commands* (in Section 5). The list of related commands at the end of most command descriptions will help identify the commands that work with one another. The example programming code in Section 6 should be helpful in showing how you can issue actual commands to the terminal from your program.

HOW THE TERMINAL'S MEMORY WORKS

The terminal contains both read-only memory and random-access memory. The read-only memory stores the firmware that supports the terminal's graphics and screen-editing features. The random-access memory stores graphics images, text, and other data used by the firmware or by your program.

There are two types of random-access memory that your program can use for specialized purposes — *graphics memory* and *program memory*. The discussion *Graphics Memory* earlier in this section, describes how your program can use graphics memory. The rest of this discussion describes how your program can use the terminal's program memory, and how the different terminal capabilities that use program memory affect each other.

Program Memory

Program memory is the terminal's general-purpose memory. It is comprised of *volatile memory* and *nonvolatile memory*.

Volatile Memory. This memory holds the terminal's input queue and dialog area buffer. Your program can adjust the size of the input queue and dialog area buffer and use the rest of this memory to define panels and to define and store macros. (The terminal uses volatile memory temporarily during panel definition, but the memory is freed once the definition is complete.)

Assuming that all settings are at their factory defaults, there are about 16,000 bytes (1000 blocks 16 bytes long) of volatile memory available to host programs, and the largest contiguous block of available memory is about 10,000 bytes.

Information stored in volatile memory is lost when the terminal is powered off or when the **FACTORY** or **RESET** commands are issued.

Nonvolatile Memory. This memory stores command settings for selected 4100-style and ANSI mode commands and for all nonvolatile macro definitions. (The terminal doesn't store this information automatically, you must issue the **SAVE NONVOLATILE PARAMETERS** command to save anything in nonvolatile memory.) The commands whose settings you can save are identified in their command descriptions in Sections 3 and 5.

The part of nonvolatile memory available for macro definitions can store about 1500 characters.

The contents of nonvolatile memory are saved even when the terminal is turned off.

Managing Program Memory

The smallest amount of program memory that the terminal allocates to any function is one memory block (16 bytes). Even if a function requires only nine bytes of memory, the terminal still uses a full memory block (16 bytes) of program memory.

The following paragraphs describe the features that use the most program memory and the restrictions that apply when you use them. The maximum settings given assume that all of the other settings are at factory default.

Dialog Area Buffer. The factory default dialog area buffer size is 49 lines. The minimum memory allocation for the Dialog area buffer is 49 lines in 80-Column mode and 30 lines in 132-Column mode. When you increase the dialog area buffer size, the amount of memory used for each additional line allocated to the dialog area buffer depends on the column mode:

- 80-Column mode requires 22 blocks (352 bytes) for the 50th line and 11 blocks (176 bytes) for each line after that.
- 132-Column mode requires 34 blocks (544 bytes) for the 31st line and 17 blocks (272 bytes) for each line after that.

Making the buffer smaller than 49 lines in 80-Column mode or 30 lines in 132-Column mode will not free any more memory space; the terminal permanently allocates that much memory for the dialog area buffer, even if it doesn't use it. However, if your dialog area buffer is larger than these minimums and you need memory space for another feature, you can retrieve some memory by making the dialog area buffer smaller.

Input Queue. The factory default input queue size is 300 bytes; you can change the queue to be larger or smaller with the SET QUEUE SIZE command. By setting the queue to less than 300 bytes, you can free more memory for use by some other feature, but settings below 108 bytes will not free any more memory.

Macro Definitions. To define a macro with the DEFINE MACRO or DEFINE NONVOLATILE MACRO command, you need volatile memory for processing the command and for storing the volatile version of the macro. The DEFINE NONVOLATILE MACRO command also requires nonvolatile memory for storing the nonvolatile version of the macro.

Each macro you define takes eight bytes for header information plus one byte for each character in its definition (a nonvolatile macro needs this much space in both volatile and nonvolatile memory). The maximum length of a macro depends on the availability of memory and on which command you use to define it. (See the discussion *Volatile and Nonvolatile Macros* earlier in this section, for information about the different commands for defining macros.)

If there is not enough program memory to define a macro (or not enough nonvolatile memory to store the nonvolatile version of a macro), the terminal issues an error message.

Panels. The terminal *temporarily* allocates memory as work space while it builds and fills a panel; the amount of memory necessary depends on the number of vertices in the panel and its orientation. Once the panel is completed, that memory again becomes available. If there is not enough volatile memory available to build a panel, your terminal will send an error message.

When You Get an Out-of-Memory Error

If the terminal runs out of program memory when you are defining a panel or a macro, the terminal sends an error message. You must free some memory if you want to continue.

- To free volatile memory, you must delete macros or reduce buffer size or queue size.
- To free nonvolatile memory, you must delete nonvolatile macros.

Reporting Program Memory Availability

You can find out how much program memory is available by sending the REPORT TERMINAL SETTINGS command from the host (or STATUS MEMORYBLOCKS from the keyboard) to query memory status. The terminal responds with a report that first gives the total number of memory blocks that remain, and then gives the number of blocks contained in the largest contiguous block of memory.

Section 5

4100-STYLE COMMANDS AND REPORTS

This section describes the terminal's 4100-style commands and reports and the types of parameters they require. The section has four parts:

- The first part describes each type of parameter used in 4100-style command syntax.
- The second part explains the conventions we've used in this manual to present command syntax.
- The third part is a dictionary-like (alphabetical) listing of all the 4100-style commands; each command description includes the command's function, syntax, and parameters, and any details unique to it.
- The fourth part describes the reports sent from the terminal to the host. It covers the parameter types used in reports, and the content of each report.

NOTE

The terminal must be in TEK mode to execute 4100-style commands sent from the host. Specify TEK mode with the SELECT CODE command, which can be issued from the host or from the keyboard.

Most 4100-style commands have two forms: one used for sending the command from the host (*host syntax*), and the other used locally from the terminal (*Setup syntax*). Host syntax uses an efficient method of packing data into a stream of ASCII characters. The packing method is described along with the explanation of parameter types. Setup syntax uses mnemonic word-forms for commands and parameters. The natural language style of Setup syntax makes it easy to enter Setup commands from the keyboard. You can enter Setup commands no matter which host command mode the terminal is in.

COMMAND HINTS

- Commands are always identified in this manual by a descriptive name in uppercase letters.
- The terminal has three command sets and four host command modes. ANSI and EDIT modes use the ANSI command set; VT52 mode uses the VT52 command set; and TEK mode uses the 4100 command set.
- A program can freely switch command modes to access the terminal's full feature set (use the SELECT CODE command, which works in all modes).
- You can issue commands from a host program (using host syntax) or from the keyboard (using Setup syntax); the host and Setup versions of a command do exactly the same thing.
- When you select Setup (by pressing the Setup key), you can issue Setup commands from the keyboard without regard to the host command mode. Setup syntax uses simple keywords and ordinary integers.
- When you issue 4100-style commands from the host, the terminal must be in TEK mode and you must encode parameter values. The different parameter types (and their encoding schemes) are explained in Section 5. Section 6 contains sample routines for encoding these parameters.
- You can save many command settings in the terminal's nonvolatile memory by issuing the SAVE NONVOLATILE PARAMETERS command; then your terminal's settings will be appropriate for your application every time you turn on the terminal.
- 4100-style commands are described at the end of Section 5. ANSI and VT52 commands are described in Section 3. Each command is described in detail, listed alphabetically by command name.
- 4100-style reports are described at the end of Section 5. ANSI and VT52 reports are described alphabetically with the command descriptions in Section 3. Section 6 contains sample routines for decoding reports.
- If you're looking for information on a specific topic, try each section's detailed Table of Contents and functional listing of commands — you'll find these on each section divider. Also try the general Table of Contents and the Index.

PARAMETER TYPES**PARAMETER TYPES**

The terminal classifies parameters according to their data format, much like some programming languages classify variable types. Parameter types fall into three categories: *host parameters*, *Setup parameters*, and *report parameters*.

Host parameters are those used in commands issued from the host. Setup parameters are those used in commands issued from the terminal keyboard. Report parameters are those that the terminal uses when sending information to the host; report parameters are explained in the discussion titled *Reports* at the end of this section.

The following discussion explains the various types of host parameters and Setup parameters. Some parameter types have simple forms, but are also used as building blocks for more complex forms such as integer arrays.

HOST PARAMETERS

Commands in host syntax use three simple parameter types: *character*, *integer*, and *xy-coordinate*. The complex variations are *string* and *integer array*.

Character Parameters

A *character parameter* is any standard character (except control characters and the D_T character). You enter the character itself.

Integer Parameters

An *integer parameter* is a sequence of up to three ASCII characters that represent the value of an integer number. You can use ASCII characters ranging from S_P (ADE 32) through D_T (ADE 127). If the value of the integer is 15 or less, you need only one character, the Lo-I character; if the value of the integer is greater than 15, you need one or two additional characters, called Hi-I characters. (Coincidentally, the ASCII characters 0 through 9 represent the encoded positive integers 0 through 9.)

NOTE

The encoding scheme that the terminal uses for integer parameter reports is different than the encoding scheme that your program must use when it issues integer parameters in 4100-style commands. The examples included here show how your program encodes integer parameters.

This discussion describes the encoded *integer parameters* that your program must use in issuing 4100-style commands from the host. Note that, when the terminal reports integer values to your program, it uses *integer report parameters*, which use a different encoding scheme. See the discussion *Integer Report Parameter* later in this section and the subroutine for decoding integer reports in *Report Decoding Subroutines* in Section 6 to see how to decode integer reports.

Figure 5-1 shows how an integer's binary form is packed into the characters that represent that integer. Notice that the integer's sign is included as a bit in the Lo-I character. If the integer is negative (as in Figure 5-1), the sign bit is 0; if the integer is positive, the sign bit is 1.

You'll probably use a routine in your program to encode integers — Section 6 contains a sample routine written in FORTRAN. However, you can manually convert numbers to encoded integers — either by either by looking them up in Appendix D (which contains a table of the encoded integers for -1049 through 1049), or by calculating them. Here's one method for calculating encoded integers:

1. Divide the absolute value of the integer by 16, reserving the quotient. Then:
 - If the integer is positive, add 48 to the remainder.
 - If the integer is negative, add 32 to the remainder.

Now you've got the ADE of the Lo-I, which is the least significant character of the parameter. If the quotient just obtained is zero, the Lo-I is the only character needed in the parameter. If the quotient is equal to or greater than 1, go on to Step 2.
2. Divide the absolute value of the quotient (reserved in Step 1) by 64, reserving the new quotient if it is equal to or greater than 1. Add 64 to the remainder to obtain the ADE of the Hi-I. If the quotient is zero, the parameter has a single Hi-I.
3. Repeat Step 2 for the reserved quotient, if any. This is the second Hi-I.

As you calculate each ADE value, look up its corresponding ASCII character and write it down from left to right, like this:

2nd Hi-I 1st Hi-I Lo-I
 (Step 3) (Step 2) (Step 1)

For example, here's how you would encode the integer + 31416.

$$\begin{aligned}
 31416 \div 16 &= 1963 && \text{Remainder of } 8 \\
 8 + 48 &= 56 && \text{Use 48 for positive integer} \\
 \text{Lo-I} &= \mathbf{8} && \text{56 is ADE of } \mathbf{8} \\
 1963 \div 64 &= 30 && \text{Remainder of } 43 \\
 43 + 64 &= 107 && \\
 \text{Hi-I} &= \mathbf{k} && \text{107 is ADE of } \mathbf{k} \\
 30 \div 64 &= 0 && \text{Remainder of } 30 \\
 30 + 64 &= 94 && \\
 \text{Hi-I} &= \mathbf{\wedge} && \text{94 is ADE of } \mathbf{\wedge}
 \end{aligned}$$

Therefore, the encoded parameter for the integer 31416 is:

\wedge k8

Here's how you would encode the integer -1024.

$$\begin{aligned}
 1024 \div 16 &= 64 && \text{Remainder of } 0 \\
 0 + 32 &= 32 && \text{Use 32 for negative integer} \\
 \text{Lo-I} &= \mathbf{s_P} && \text{32 is ADE of } \mathbf{s_P} \\
 64 \div 64 &= 1 && \text{Remainder of } 0 \\
 0 + 64 &= 64 && \text{64 is ADE of } \mathbf{@} \\
 \text{Hi-I} &= \mathbf{@} && \\
 1 \div 64 &= 0 && \text{Remainder of } 1 \\
 1 + 64 &= 65 && \text{65 is ADE of } \mathbf{A} \\
 \text{Hi-I} &= \mathbf{A} &&
 \end{aligned}$$

Therefore, the encoded parameter for the integer -1024 is:

A@s_P

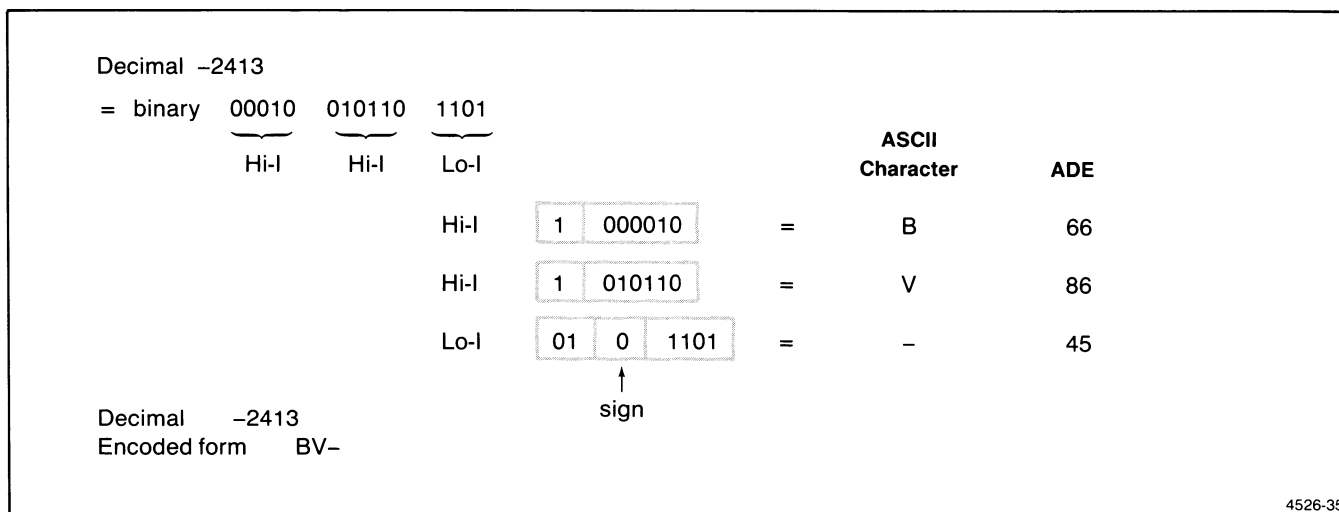


Figure 5-1. How to Encode Integer Parameters.

XY-Coordinate Parameters

An *xy-coordinate parameter* is a sequence of up to five ASCII characters. This sequence represents the numerical values of both the x- and y-coordinates used to specify locations for graphics operations. Figure 5-2 shows the bit-packing scheme for each character, what each character means in the sequence (Hi-Y, Extra, etc.), and the order in which you must send the characters (begin with Hi-Y and end with Lo-X).

Here again, you'll probably use a routine in your program to encode xy-coordinates for transmission to the terminal — see Section 6 for a FORTRAN example of such a routine. You can, however, manually calculate the ADE of each character in the encoded sequence; here's a method to follow:

1. Divide *y* by 128 and discard the remainder (if any). Add 32 to the integer quotient. The sum is the ADE of the Hi-Y character.
 You can omit the Hi-Y character if it is the same as the last Hi-Y sent.
2. Find the Extra character.
 - a. Divide *y* by 4. The remainder is the first half of the Extra.
 - b. Divide *x* by 4. The remainder is the second half of the Extra.
 - c. Multiply the first half of the Extra by 4. To the product add the second half of the Extra and 96. This sum is the ADE of the Extra character.
3. Divide *y* by 4 and discard the remainder. Divide the integer quotient by 32 and add 96 to the remainder. The sum is the ADE of the Lo-Y character.
4. Divide *x* by 128 and discard the remainder (if any). Add 32 to the integer quotient. The sum is the ADE value of the Hi-X character.
5. Divide *x* by 4 and discard the remainder (if any). Divide the integer quotient by 32 and add 64 to the remainder. The sum is the ADE of the Lo-X character.

Here's how you would encode the xy-coordinate 53,1000 (See Figure 5-2).

$1000 \div 128 = 7$	Discard remainder of 104
$7 + 32 = 39$	39 is ADE of 'I'
$Hi-Y = 'I'$	
$1000 \div 4 = 250$	Remainder of 0
$0 * 4 = 0$	
$First\ Extra = 0$	
$53 \div 4 = 13$	Remainder of 1
$Second\ Extra = 1$	
$0 + 1 + 96 = 97$	97 is ADE of 'a'
$Extra = a$	
$1000 \div 4 = 250$	Remainder of 0
$250 \div 32 = 7$	Remainder of 26
$26 + 96 = 122$	122 is ADE of 'z'
$Lo-Y = z$	
$53 \div 128 = 0$	Remainder of 53
$0 + 32 = 32$	32 is ADE of 'P'
$Hi-X = P$	
$53 \div 4 = 13$	Discard remainder of 1
$13 \div 32 = 0$	Remainder of 13
$13 + 64 = 77$	77 is ADE of 'M'
$Lo-X = M$	

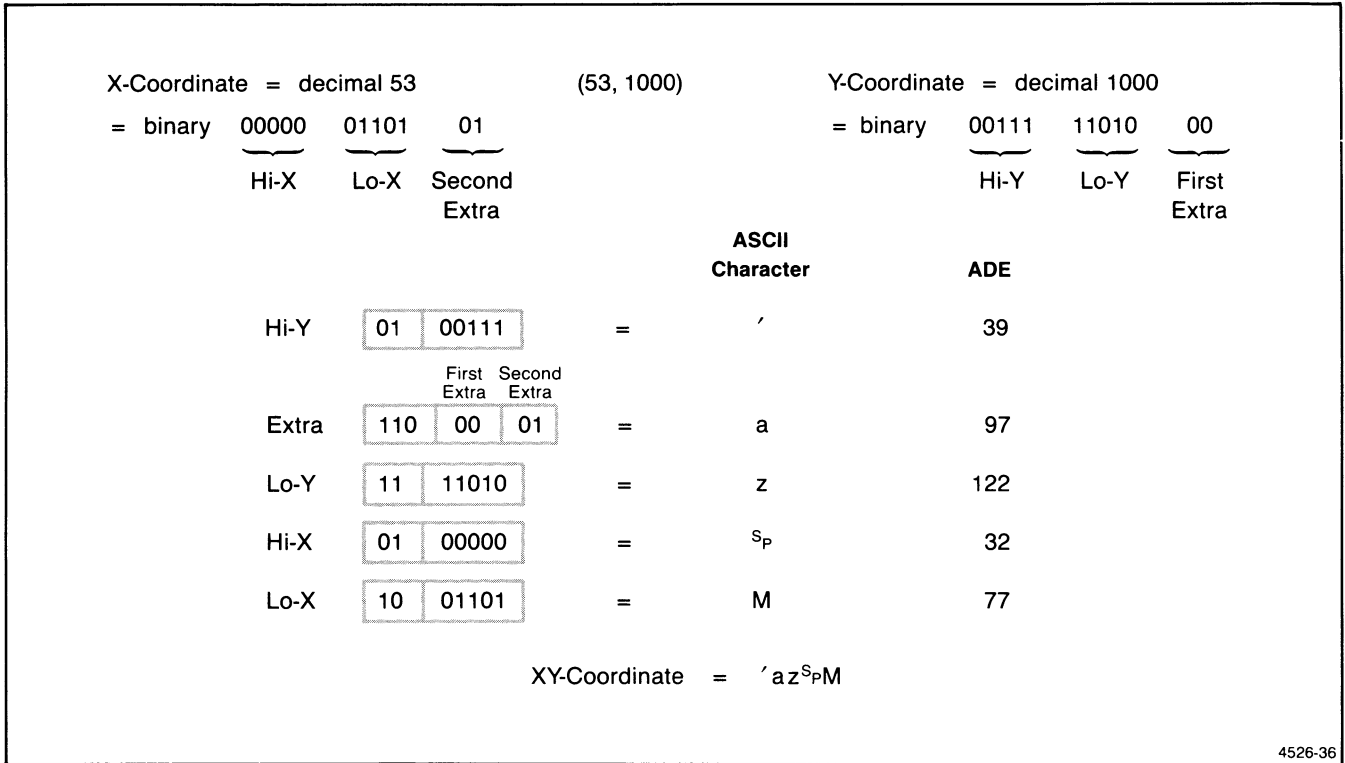
Therefore, the encoded xy-coordinate parameter 53,100 is:

'az^sP^rM

Since the Lo-X character terminates the xy-coordinate sequence, it is the only one that must always be sent. You can omit any or all of the other characters, but only under certain circumstances.

Figure 5-3 shows the syntax of the xy-coordinate parameter and indicates the circumstances under which you can omit characters. For example:

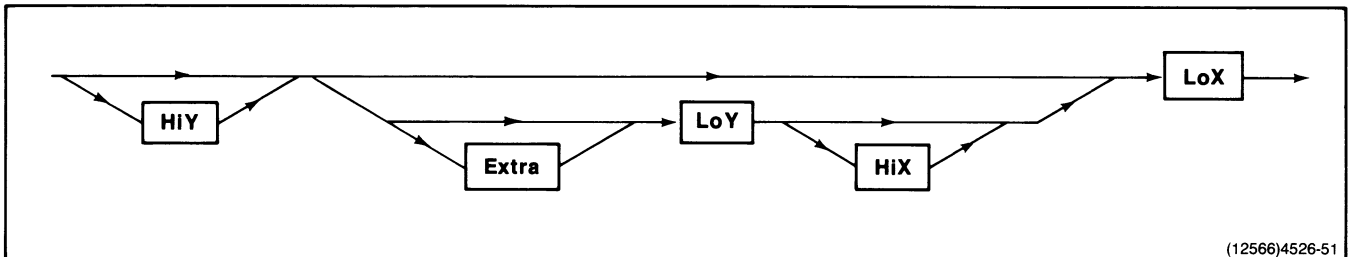
- You can omit the Hi-Y character if it is the same as the last Hi-Y character that you sent.
- You can omit Lo-Y only if you also omit Hi-X, or to put it another way, if you send Hi-X, you must precede it with Lo-Y.
- You can omit the Extra character if 10-bit resolution is adequate for your purposes. If you include the Extra character, the Lo-Y character must follow it.



XY-Coordinate = 'az^SpM

4526-36

Figure 5-2. How to Encode XY-Coordinates.



(12566)4526-51

Figure 5-3. XY-Coordinate Parameter Syntax.

PARAMETER TYPES**String Parameters**

A *string* is a group of ASCII characters sent to the terminal as a single parameter. You must precede the characters that make up the string by a *count*, an encoded integer that tells the terminal how many characters are in the string. Encode the integer as described earlier under *Integer Parameters*. If, for example, you want to send the phrase *PRESS RETURN KEY*, you would send the following sequence:

```
A0PRESSSPRETURNSPKEY
```

The *A0* (the encoded integer for 16) at the beginning of the string is the count that tells the terminal that 16 characters (including spaces) follow.

Integer Array Parameters

An *integer array* consists of a sequence of integer parameters, all encoded as described earlier. The first integer in the sequence is the count that tells the terminal how many integers follow, with subsequent integers forming the individual elements of the array.

For example, to send the integers *1*, *5*, *-1*, and *16* as an array of four integers, you would send the following:

```
415!A0
```

The *4* at the beginning of the array is the count that tells the terminal that four integers (not four characters) follow. Each of the integers is represented by its encoded value: *4*, *1*, and *5* are the encoded values of the integers 4, 1, and 5; *!* is the encoded value for -1; and *A0* is the encoded value of 16.

SETUP PARAMETERS

Parameters in Setup syntax are similar to those in host syntax; their differences are explained here. Generally speaking, Setup syntax parameter types are simpler, but there are more of them. They are: *character*, *integer*, *small integer*, *xy-coordinate*, *keyword*, and *key specifier*. There are three complex variations: *integer array*, *string*, and *real parameter*.

Character Parameters

A *character parameter* in Setup syntax is an ASCII printing character in the range *SP* to *~* (tilde), that is, ADE 32 through 126. The alphanumeric and symbol keys on the keyboard fall within this range. When you specify a character parameter in Setup syntax, you can type either the actual character or its ADE value.

Integer Parameters

An *integer parameter* in Setup syntax is simply the decimal number itself (you don't encode it like you do in host syntax). For example, to set both the transmit and receive rate to 2400 using the BAUDRATE Setup command, enter:

```
BAUDRATE 2400,2400
```

Small Integer Parameters

A *small integer parameter* ranges from 0 through 127, which is the ADE range of characters *NV* through *DT*. When you specify a small integer parameter, you can enter either the actual character or its ADE value.

To use an ADE value in the range 0 through 9, you must precede the digit with 0. For example, to use the ADE value for *HT* (ADE 9), you enter *09*. Otherwise, the single digits 0 through 9 specify the digit characters (ADE 48 through ADE 57).

XY-Coordinate Parameters

An *xy-coordinate parameter* in Setup syntax is simply the decimal values of *x* and *y*. For example, to issue the MOVE command with the coordinates 500,500, you would enter:

```
MOVE 500,500
```

Keyword Parameters

A *keyword parameter* specifies what action you want a command to perform. You can spell out the entire keyword or enter just as many characters as necessary to distinguish that keyword from other keywords used for that command. For example, to issue the SET GRAPHTEXT CHARACTER PATH command to cause graphtext to write from top to bottom, you could enter the Setup command:

```
GTPATH DOWN
```

Or just:

```
GTPATH D
```

Key Specifier Parameters

A *key specifier* parameter type is used in Setup in the DEFINE MACRO and DEFINE NONVOLATILE MACRO commands to specify which key will receive a macro definition. You enter the key specifier by pressing the key itself or by typing a label that identifies the key. See the discussion of *Key Macros* in the DEFINE MACRO command description for more details.

Integer Array Parameters

An *integer array* in Setup syntax is like the host syntax integer array in that it consists of a sequence of integer parameters. Unlike host syntax, however, you don't include a count at the beginning of the array, and you don't encode the integers. Rather, you simply enter the value of each item in the array, separating each item with a space or a comma.

String and Delimited String Parameters

A *string* parameter consists of a group of any alphanumeric or symbol characters on the terminal keyboard. Enter the actual characters, rather than ADE values.

Some commands require a *delimited string*, which simply means you must use a special character before and after the string to distinguish that group of characters from others in the command. The delimiter can be any character, except a comma, a space, or an edit character (see the SET EDIT CHARACTERS command for an explanation of edit characters). The opening and closing delimiters must be the same character, but cannot be a character that is also in the string. Thus, a delimited string consists of the opening delimiter, a string of keyboard characters, and the closing delimiter.

For example, in the Setup syntax for the SET EOF STRING command, you specify up to ten characters to indicate the end-of-file marker. So, to set the end-of-file marker to be the characters *THE END*, you could enter:

```
EOFSTRING /THE END/
```

In this example, we've used the slash character (/) as the delimiter.

COMMAND CONVENTIONS

All 4100-style command descriptions are consistently structured, using an easy-to-read set of syntax conventions. Figure 5-4 and the following discussion give a summary of the overall structure of a command description and of the notation used to show syntax.

Here are the conventions used to represent host and Setup syntax:

- Characters shown in bold type are those you must enter exactly as shown.
- Most 4100-style commands have both host syntax and Setup syntax. As you can see in Figure 5-4, the host syntax and Setup syntax appear in separate boxes:
 - The *Host Syntax* box shows the way a host application would send this command to a terminal.
 - The *Setup Syntax* box shows the way you would enter this command at a terminal keyboard.
- Parameter names are shown on separate lines to make the syntax easier to read. However, when entering commands, follow these rules:
 - In Setup syntax, enter all parts of a command on the same line. The first character after the command name must be a space; use one or more spaces or a comma to separate parameters.
 - In host syntax, issue the E_c character (if required), the command's opcode, and any parameters. Do not use spaces between any characters (except Space characters that are part of an encoded parameter).
- You can abbreviate the Setup name — just enter as many letters of the name as are needed to identify it uniquely. In the example in Figure 5-4, the Setup name *DALINES* can be abbreviated *DAL* (if you tried to abbreviate this to *DA*, the terminal would issue an error message since it wouldn't know whether you want to issue the *DALINES* command or another command, such as *DAINDEX*).
- Many command descriptions include examples showing how to use the command. When both host and Setup examples are included, the two examples do the same thing.

Individual descriptions of each parameter follow the syntax boxes. A parameter description includes the parameter type, the range of valid values, and the default values. Each parameter has up to two types of defaults:

- *Factory* — The value assigned a parameter when the terminal is shipped from Tektronix. You can issue the **FACTORY** command to restore parameters to this value.
- *Omitted* — The value assigned a parameter if the Setup command is used and no value is specified for the parameter. This value is used only in Setup syntax (see *About Omitting Parameters*).

Any additional explanation follows the parameter descriptions. Parameter names always appear in italics.

Many command descriptions show a typical example of the command in both host syntax and Setup syntax. Both the host example and the Setup example use the same parameter values, and thereby perform the same action.

Most of the command descriptions include a list of related 4100-style commands. For a few command descriptions, the list of related commands includes ANSI commands; these are described in Section 3. The commands in the list may affect or be affected by the command described, or they may perform a similar function. You should read the descriptions of related commands to gain a more complete idea of how a specific feature works.

SET DIALOG AREA LINES

Specifies the number of lines visible in the dialog area.

Host Syntax

```
EscLL number-of-lines
```

Setup Syntax

```
DALINES number-of-lines
```

number-of-lines: integer; specifies how many lines are in the dialog area. Must be in the range 2 through 30.

Defaults Factory = 30
Omitted = Error

If you make the dialog area larger than the dialog buffer (assuming both are less than 30 lines), the terminal expands the dialog buffer to be as large as the dialog area.

Syntax Example

```
Host: EscLL?  
Setup: DALINES 15
```

Sets the dialog area to 15 lines (encoded ?).

Related Commands:

```
SET DIALOG AREA BUFFER SIZE  
SET DIALOG AREA VISIBILITY
```

4526-52

A SAMPLE COMMAND DESCRIPTION

Figure 5-4 uses the SET DIALOG AREA LINES command to illustrate this section's command description format:

- *Host Syntax*. To issue this command from the host, send the escape sequence EscLL, followed by the *number-of-lines* parameter.
- *Setup Syntax*. To issue this command from the keyboard, type the Setup name, **DALINES**, followed by the *number-of-lines* parameter and a carriage return.
- *Parameter*. There is a brief description of what the *number-of-lines* parameter does.
- *Default*. The *number-of-lines* parameter defaults to 30 lines; if you omit this parameter the terminal sends an error message.
- *Example*. The example shows how to issue this command from the host and from Setup.
- *Related Commands*. This lists other commands that affect (or are affected by) the SET DIALOG AREA LINES command.

Figure 5-4. A Typical 4100-Style Command Description.

ABOUT OMITTING PARAMETERS

As a general rule, you cannot omit parameters when issuing commands from the host; you can omit parameters in Setup as long as the terminal's defaults will accomplish what you want:

- In host syntax, all of the command's parameters must be included for the terminal to execute the command properly. Because the terminal expects to receive parameters in a specified sequence from the host, you cannot skip one parameter and supply the next.

The terminal processes a host syntax command when it receives any of the following *command terminators*:

- The ^c character, which starts a new command
- The ^o character, which instructs the terminal to send its answerback string to the host
- The ^s , ^g , or ^u characters, which instruct the terminal to enter one of the implicit command modes (Marker, Vector, or Alpha modes, respectively)

When the terminal receives one of these command terminators before it receives a complete list of parameter values, it assigns a value of 0 to the omitted parameters. Thus, omitting parameters is the same as assigning them a value of 0 — and if 0 is not a valid value for the parameters, the terminal will detect an error.

- In Setup syntax, you can omit parameters from most commands and the terminal will supply a default value. If the parameter is the only one in the command or is the last of two or more parameters, you simply omit it. To omit a parameter other than the last one, use commas to separate the omitted parameter's location from adjacent parameters. For instance, to omit the first parameter of the SET DIALOG AREA INDEX command, you enter:

```
DAINDEX ,2,3
```

To omit the second parameter you enter:

```
DAINDEX 1,,3
```

SAVING COMMAND SETTINGS

You can save the settings of some commands by issuing the SAVE NONVOLATILE PARAMETERS command before you turn off the terminal. Then the terminal will retain these settings in its nonvolatile memory even when it is powered off. The commands that you can save are identified following the command's statement of purpose with the phrase *Can be saved in nonvolatile memory*. There is an alphabetic listing of commands that can be saved in the *4105 Computer Display Terminal Programmers Reference Guide*. As Figure 5-4 shows, SET DIALOG AREA LINES is one of those commands.

MORE INFORMATION ABOUT COMMANDS

This manual and the *4105 Computer Display Terminal Reference Guide* provide summary information about the commands in several places:

- The divider tabs for this section lists the 4100-style commands by functional groupings.
- Appendix C of this manual contains command summary tables. By scanning this appendix, you can, for example, see all the parameter defaults for all commands at a glance. You can find the Setup name of a command for which you know the descriptive name, and you can quickly find a command's opcode or what its parameters are.
- The *4105 Computer Display Terminal Programmers Reference Guide* contains additional cross reference lists, which list commands by Setup name and by escape sequence.

4100-STYLE COMMAND DESCRIPTIONS

This part of Section 5 contains descriptions of the terminal's 4100-style commands. When you issue these commands from the host, you'll need to encode the parameters, using encoding schemes discussed earlier in this section. The commands are presented alphabetically according to their descriptive names. Section 4, *Graphics Concepts*, describes how these 4100-style commands work together.

BEGIN PANEL BOUNDARY

Starts the definition of a panel boundary.

Host Syntax

```
^cLP first-point
      draw-boundary
```

Setup Syntax

```
BEGINPANEL first-point
            draw-boundary
```

first-point: xy-coordinate; indicates the first point in a panel boundary. The valid range of values is 0 through 4095 for both the x- and y-coordinate.

Defaults: Factory = (none)
 Omitted = 0,0

draw-boundary: integer; specifies whether the fill pattern covers the panel boundary. Must be one of the following:

- 0 The fill pattern covers the panel boundary
- 1 The boundary is displayed around the finished panel, using the current line style and line index

Defaults: Factory = (none)
 Omitted = 0

After issuing BEGIN PANEL BOUNDARY, you can define the boundary of the panel in either of two ways:

- Implicitly, using Vector or Marker mode.
- Explicitly, using MOVE and DRAW commands.

During a panel definition, the terminal interprets xy-coordinates as the vertices of the panel's boundary-line. This means that you cannot draw a marker during a panel definition, although you can define a panel when the terminal is in Marker mode.

You don't need to draw the panel's last boundary line explicitly. When the terminal receives the END PANEL command, it closes the panel and fills it with the fill pattern specified in the SELECT FILL PATTERN command. Appendix F displays all of the terminal's predefined fill patterns.

Multiple Panel Boundaries. You can create a panel with multiple boundaries, as shown in Figure 5-5 (next page). To do this, send the BEGIN PANEL BOUNDARY command to define the first panel boundary. When you want to start the second boundary, send another BEGIN PANEL BOUNDARY command. Don't use the END PANEL command to close the first boundary — the second BEGIN PANEL BOUNDARY command closes the first boundary and starts another boundary at the specified position. When you issue END PANEL, the last boundary is closed and the entire panel (as defined by the multiple boundaries) is filled.

Defining panels uses volatile memory that could be used for other features such as the input queue or macro definitions. See the discussion *Managing Program Memory* in Section 4 for an explanation of how panel definitions affect the availability of volatile memory.

Syntax Example

```
Host: ^cLP ^az^PM1
Setup: BEGINPANEL 53,1000,1
```

Starts a panel definition at 53,1000 (the xy-coordinate 53,1000 is encoded ^az^PM).

Related Commands

```
END PANEL
ENTER MARKER MODE
ENTER VECTOR MODE
SELECT FILL PATTERN
SET LINE INDEX
SET LINE STYLE
```

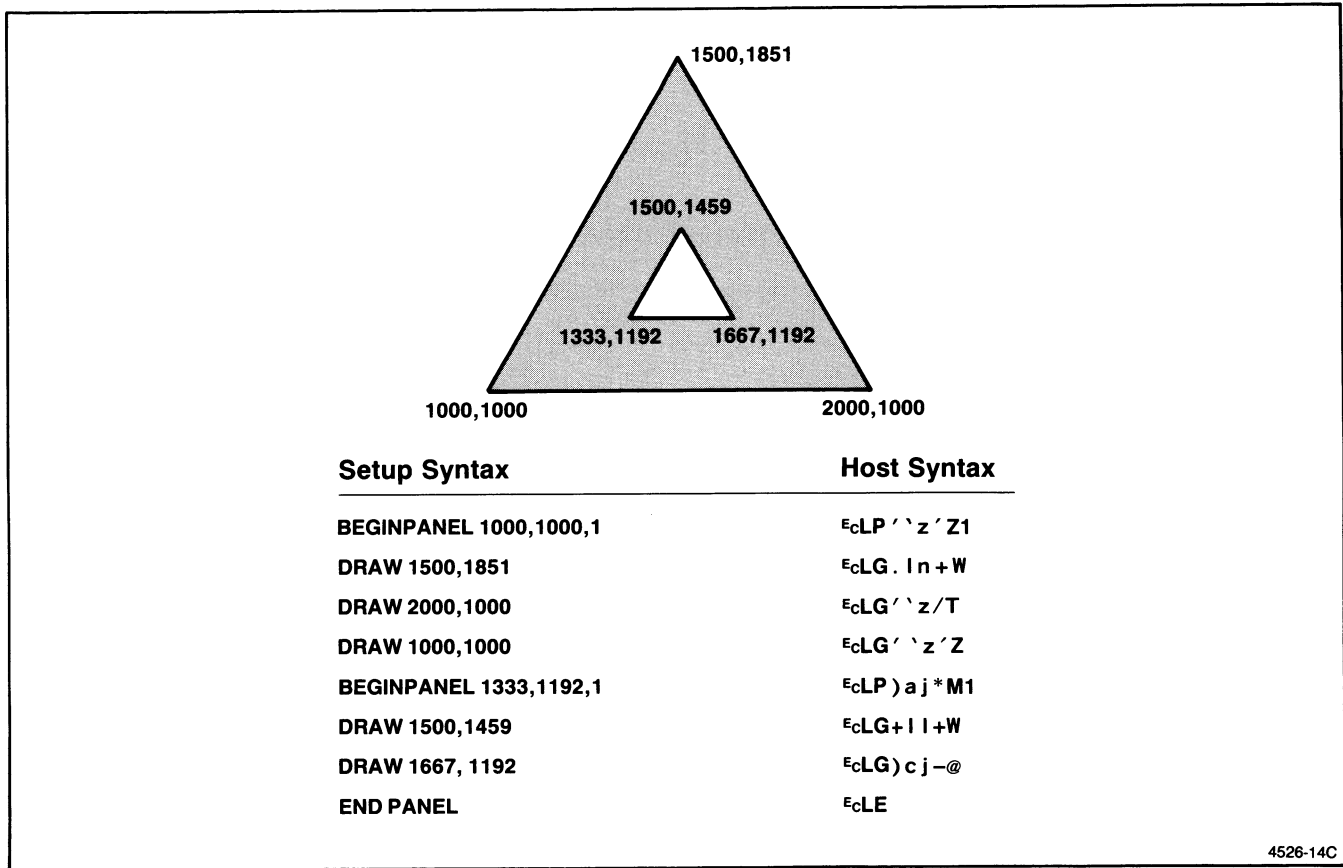


Figure 5-5. Creating a Panel with Multiple Boundaries.

BEGIN PIXEL OPERATIONS

Sets up the terminal for subsequent pixel operations.

Host Syntax

```

EcRU surface-number
    ALU-mode
    bits-per-pixel
  
```

Setup Syntax

```

PXBEGIN surface-number
    ALU-mode
    bits-per-pixel
  
```

surface-number: integer; specifies the surface on which subsequent pixel commands will write (or read) data. The values -1, 0, and 1 are valid, but since Surface 1 is the terminal's only surface, all valid values select Surface 1. (Other Tektronix terminals can display more than one surface, so the surface parameter is included here for compatibility.)

Defaults: Factory = 1
Omitted = 0

ALU-mode: integer; specifies the writing mode. Valid values are 0, 7, 11, 12, or 15 — Table 5-1 describes the function that each value selects.

Defaults: Factory = 11
Omitted = 0

bits-per-pixel: integer; specifies the number of bits used to encode the color index for each pixel in subsequent RASTER WRITE and RUNLENGTH WRITE commands. Valid values are 0, 1, 2, 3, 4, and 6; 0 means no change.

Defaults: Factory = 6
Omitted = 0

This command sets values used in these pixel commands: RASTER WRITE, RUNLENGTH WRITE, RECTANGLE FILL, and PIXEL COPY.

See *Manipulating Pixels* in Section 4 for a complete explanation of how to work with pixels.

Syntax Example

```

Host: EcRU1 < 6
Setup: PXBEGIN 1,12,6
  
```

Begins pixel operations with ALU mode set to *AND* (selected with integer 12, encoded <) and six bits per pixel.

Related Commands

```

PIXEL COPY
RASTER WRITE
RECTANGLE FILL
RUNLENGTH WRITE
  
```

Table 5-1
ALU MODES

ALU Mode	Function	Operation
0	No change	
7	A XOR B (exclusive OR)	Pixels being written (Index B) are logically XORed with pixels in existing image (Index A) to form the new pixels to be displayed, and the pixel image can be erased by writing image again
11	B	Pixels being written replace pixels in existing image (default <i>ALU-mode</i>)
12	A AND B	Pixels being written (Index B) are logically ANDed with pixels in existing image (Index A) and the results are displayed
15	A OR B	Pixels being written (Index B) are logically ORed with pixels in existing image (Index A) and the results are displayed

CANCEL**CANCEL**

Stops terminal activity and resets several terminal settings to their default values.

Host Syntax

```
EcKC
```

Setup Syntax

```
CANCEL
```

When you issue this command (which has the same effect as pressing the Cancel key) it:

- Puts the terminal in Alpha mode and terminates all of the following:
 - Vector mode
 - Marker mode
 - Bypass mode
 - Prompt mode
 - Snoopy mode
 - GIN
- Unlocks the keyboard (see the LOCK KEYBOARD command).
- Terminates any copy or hard copy function currently in progress; see the 4100-style commands COPY and HARDCOPY and the ANSI command MC (MEDIA COPY).
- Flushes input and output queues (see the SET QUEUE SIZE command). Characters not yet sent to the host will be discarded and ignored.

Related Commands

```
COPY
ENTER ALPHA MODE
ENTER BYPASS MODE
ENTER MARKER MODE
ENTER VECTOR MODE
HARDCOPY
LOCK KEYBOARD
PROMPT MODE
SET ALPHATEXT FONT
SET QUEUE SIZE
SET SNOOPY MODE
```

CLEAR DIALOG SCROLL

Erases the dialog buffer.

Host Syntax

```
EcLZ
```

Setup Syntax

```
CLEARDIALOG
```

Issuing CLEAR DIALOG SCROLL has the same effect as pressing the terminal's D Eras key.

After the dialog buffer is cleared, the cursor returns to the home position (upper-left corner of dialog area buffer).

COPY

Sends data from the host directly to a copier or printer.

Host Syntax

```

EcJC source
  separator
  destination

```

Setup Syntax

```
COPY HO: TO destination
```

source: string; specifies the data source, which is always the host. Must be the string *HO*: (in uppercase or lowercase).

Defaults: Factory = (none)
Omitted = Error

separator: string; separates the source and destination parameters. It may be omitted in Setup syntax or be an empty string in host syntax. If included, must be the string *TO* (in uppercase or lowercase).

Defaults: Factory = (none)
Omitted = Error

destination: string; specifies the destination port. Must be *HC*:, the COPIER port.

Defaults: Factory = (none)
Omitted = Error

NOTE

You can issue the COPY command in Setup, but it is not recommended practice. If the data you copy is not terminated by an EOF string, subsequent commands and data will also be sent to the copier instead of to the terminal. You can press the Cancel key in this case to end the copy and return control to the terminal.

When you issue this command, the terminal passes all data it receives from the host directly to a copier or printer. The data is not processed by the terminal and is not displayed on the terminal screen. The host program is responsible for structuring data so that the copier or printer can use it.

The host must include an end-of-file string at the end of the data, and the terminal must know what end-of-file string to expect from the host. Use the SET EOF STRING command to set the terminal's end-of-file string to be the same as the one the host sends.

The copy terminates when the terminal receives an end-of-file string from the host or when the user presses the Cancel key.

Before sending a copy, use the SELECT HARDCOPY INTERFACE command to specify the copy device you are using. For graphics or dialog copies, the device can be a Tektronix 4691, 4692, or 4695 Color Graphics Copier, a Tektronix 4644 Dot Matrix Printer, a Hewlett-Packard ThinkJet, or another copier that uses Epson-style graphics. You can make dialog copies on some Centronics-style printers that lack graphics capability.

Syntax Example

```
Host: EcJC3HO:2TO3HC:
```

Copies data from the host to a copier or printer.

Related Commands

```

SELECT HARDCOPY INTERFACE
SET EOF STRING
SET EOL STRING
SET HARDCOPY MONOCHROME ATTRIBUTES

```

CRLF**CRLF**

Specifies whether a C_R character sent to the terminal also implies a L_F character. (Can be saved in nonvolatile memory.)

Host Syntax

```
 $E_c$ KR crlf-mode
```

Setup Syntax

```
CRLF crlf-mode
```

crlf-mode: integer (keyword in Setup); must be one of the following:

Host	Setup	
0	no	C_R does not imply L_F
1	yes	C_R implies L_F

Defaults: Factory = 0
Omitted = 1

When C_R implies $C_R L_F$, the terminal performs a Carriage Return and Line Feed combination. The cursor moves to the beginning of the *next* line on the display.

When C_R does not imply $C_R L_F$, a C_R moves the cursor to the beginning of the *current* line, not the next line.

The Carriage Return and Line Feed combination only affects a C_R sent to the terminal screen. That is, when you press the Return key, the implied L_F character is not sent to the host.

DEFINE MACRO

Creates or deletes a volatile macro.

Host Syntax

```
 $E_c$ KD macro-number  
macro-contents
```

Setup Syntax

```
DEFINE macro-number  
string
```

macro-number: integer (key specifier or integer in Setup); specifies the number of the macro being defined. Valid values range from -150 through 32767 (except -1).

Specifying -1 or the keyword *all* deletes all volatile macros.

Defaults: Factory = (none)
Omitted = 0

macro-contents: integer array; specifies ADEs that represent the characters defining the macro. Each integer in the array must be in the range 0 through 127. (Host syntax only.)

Defaults: Factory = (none)
Omitted = Empty array

string: delimited string; defines the macro. The string must consist of characters whose ADEs are in the range 0 through 127. (Setup syntax only.)

Defaults: Factory = (none)
Omitted = Empty string

If you're defining a macro in Setup syntax, you must precede a C_R or any special editing characters in the macro definition with the *literal character* (see the SET EDIT CHARACTERS command for further explanation).

After a macro is defined, you can expand it either from the host with the EXPAND MACRO command (for any macro) or from the keyboard by pressing the key that corresponds to the macro number.

Volatile and Nonvolatile Macros. The discussion *Volatile and Nonvolatile Macros* in Section 4 explains the difference between these two. The DEFINE MACRO command defines and deletes only volatile macros. To define or delete a nonvolatile macro, use the DEFINE NONVOLATILE MACRO command.

Deleting Macros. To delete a macro in either host or Setup syntax, you issue the DEFINE MACRO command with the macro's number, but without the macro definition. In Setup syntax, don't include the *string* parameter. In host syntax, use 0 as the array count for *macro-contents*. To delete all macros, specify -1 (or the keyword *all* in Setup) for *macro-number*.

Key Macros. The keyboard macro charts in Appendix A show the macro numbers assigned to the terminal's keys. Note that the macro number for most keys is the ADE of the character that the key normally produces.

As shown in the keyboard macro charts, each key is associated with up to four macro numbers: unshifted, shifted, Ctrl, and Ctrl-shifted (in some cases, the unshifted and shifted positions of a key generate the same macro).

Note that when you define a key in Setup you can either enter the key's ASCII decimal equivalent (ADE) or its *key specifier*, which means that you just press the key (provided this key normally produces an ASCII character). For the function keys (which don't produce ASCII characters), just enter the label printed on the key; for the shifted versions use the specifier S1 — S8. For example, to define a macro for the key F5, enter the characters *F* and *5* for the *macro-number* parameter; to define a macro for the shifted version of F5, enter *S* and *5*. For other non-ASCII keys, such as the Joydisk, you must enter the key's macro number.

Note, too, that you enter the characters in the macro's definition as a delimited string rather than an array of ADEs.

When you press a key to expand a key macro, the terminal sends the macro's contents to the host, unless the macro includes a special character called the *key execute character*, which executes the contents locally (refer to *Keeping a Key Macro Local* in Section 4).

Defining and storing macros uses program memory that could be used for other features such as the input queue or panel definitions. See the discussion *Managing Program Memory* in Section 4 for an explanation of how macro definitions affect the availability of program memory.

Syntax Example

Host: `^cKDH03E8E9E:`
Setup: `DEFINE F1,/XYZ/`

Defines a macro to make the F1 key (Macro 128, encoded *H0*) generate the character string *XYZ*. In the host example, *X* has ADE 88, encoded *E8*; *Y* has ADE 89, encoded *E9*; *Z* has ADE 90, encoded *E:*.

Related Commands

EXPAND MACRO
DEFINE NONVOLATILE MACRO
LEARN
LEARN NONVOLATILE
MACRO STATUS

DEFINE NONVOLATILE MACRO**DEFINE NONVOLATILE MACRO**

Creates or deletes both the volatile and nonvolatile versions of a macro. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcKO macro-number
      macro-contents
```

Setup Syntax

```
NVDEFINE macro-number
           string
```

macro-number: integer (key specifier or integer in Setup); specifies the number of the macro being defined. Valid values range from -150 through 32767 (except -1). Specifying -1 or the keyword *all* deletes all nonvolatile macros.

Defaults: Factory = (none)
Omitted = 0

macro-contents: integer array; specifies ADEs that represent the characters defining the macro. Each integer in the array must be in the range 0 through 127. (Host syntax only.)

Defaults: Factory = (none)
Omitted = Empty array

string: delimited string; defines the macro. The string must consist of characters whose ADEs are in the range 0 through 127. (Setup syntax only.)

Defaults: Factory = (none)
Omitted = Empty string

This command works just like the **DEFINE MACRO** command; the only distinction between these two commands is that you can save a macro defined with the **DEFINE NONVOLATILE MACRO** command in the terminal's nonvolatile memory.

NOTE

*To actually save or delete a macro in nonvolatile memory, you must issue the **SAVE NONVOLATILE PARAMETERS** command before the terminal is reset or turned off.*

Syntax Example

```
Host: EcKOH03E8E9E:
      EcKU
Setup: NVDEFINE F1,/XYZ/
       NVSAVE
```

Defines and saves a macro that programs the F1 key (Macro 128, encoded *H0*) to generate the character string XYZ. In the host example, 3 is the array count, X is ADE 88, encoded *E8*; Y is ADE 89, encoded *E9*; Z is ADE 90, encoded *E*.

Related Commands

```
DEFINE MACRO
EXPAND MACRO
LEARN
LEARN NONVOLATILE
MACRO STATUS
SAVE NONVOLATILE PARAMETERS
```


DRAW

Draws a vector from the current graphics position to a new position.

Host Syntax

```
EcLG position
```

Setup Syntax

```
DRAW position
```

position: xy-coordinate; indicates the point to draw to. The valid range of values is 0 through 4095 for both the x- and y-coordinates.

Defaults: Factory = (none)
Omitted = 0,0

When you issue this command, the terminal draws a vector in the line style and line index set by the SET LINE STYLE and the SET LINE INDEX commands.

The DRAW command has the same effect as sending the terminal an xy-coordinate when the terminal is in Vector mode.

If the terminal is in Alpha, Vector, or Marker mode when it receives the DRAW command, it stays in that mode.

See *Creating Images with Graphics Primitives* in Section 4 for more information about creating and using lines.

Syntax Example

```
Host: EcLG 'azSPM  
Setup: DRAW 53,1000
```

Draws a vector from the current graphics position to 53,1000 (the xy-coordinate 53,1000 is encoded 'azSPM).

Related Commands

```
ENTER VECTOR MODE  
MOVE  
SET LINE INDEX  
SET LINE STYLE
```

DRAW MARKER

Draws a marker at a specified location.

Host Syntax

```
EcLH position
```

Setup Syntax

```
MARKER position
```

marker-position: xy-coordinate; specifies where you want the marker drawn. Valid range of values is 0 through 4095 for both the x- and y-coordinate.

Defaults: Factory = (none)
Omitted = 0,0

When you issue this command, the terminal draws a marker at the specified location in the style and color index specified by the SET MARKER TYPE and SET LINE INDEX commands.

The DRAW MARKER command has the same effect as sending the terminal an xy-coordinate when the terminal is in Marker mode.

If the terminal is in Vector, Alpha, or Marker mode when it receives the DRAW MARKER command, it stays in that mode.

The ALU mode specified by the SET GRAPHICS AREA WRITING MODE command affects the way the terminal draws markers on the screen. See the SET GRAPHICS AREA WRITING MODE command for details about how the ALU modes operate.

Syntax Example

```
Host: EcLH 'azSPM  
Setup: MARKER 53,1000
```

Draws a marker at 53,1000 (the xy-coordinate 53,1000 is encoded 'azSPM).

Related Commands

```
DRAW  
ENTER MARKER MODE  
ENTER VECTOR MODE  
SET LINE INDEX  
SET MARKER TYPE
```

ENABLE DIALOG AREA

ENABLE DIALOG AREA

Enables or disables the dialog area. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcKA mode
```

Setup Syntax

```
DAENABLE mode
```

mode: integer (keyword in Setup); specifies whether the dialog area is enabled. Must be one of the following:

Host Setup

- 0 no Disables the dialog area
- 1 yes Enables the dialog area

Defaults: Factory = 1
Omitted = 1

When the dialog area is *enabled* and the terminal is in TEK mode, all alphatext is directed to the dialog buffer (for the text to be seen, the dialog area must also be visible; see the SET DIALOG AREA VISIBILITY command).

When the dialog area is *disabled* and the terminal is in TEK mode, the terminal displays alphatext at the current graphics position in the graphics area. This is how to emulate Tektronix 4010 Series terminals, which do not have a dialog area.

In ANSI, EDIT, or VT52 modes, the terminal automatically directs text to the dialog area, regardless of whether the dialog area is enabled.

While the terminal is in Setup, all text entered from the keyboard is displayed in the dialog area.

Table 5-2 summarizes the effects of enabling and disabling the dialog area.

Related Commands

- ENABLE 4010 GIN
- ENTER ALPHA MODE
- PAGE
- SET DIALOG AREA VISIBILITY
- SET LINE STYLE

Table 5-2
EFFECTS OF ENABLE DIALOG AREA

Feature	Dialog Area Disabled	Dialog Area Enabled
Alphatext	Sent to the current graphics position in the graphics area	Sent to the current alpha cursor position in dialog area
G Eras Key S Eras Key PAGE Command	Erases the graphics area of the screen (S Eras also erases the dialog area) Takes the terminal out of GIN Resets the terminal to Line Style 0 Sets current position to the home position (0,3071) Puts terminal in Alpha mode	Erases the graphics area (S Eras also erases the dialog area)
Cr Character	Puts terminal in Alpha mode Performs a carriage return action Resets the terminal Line Style to 0 Takes the terminal out of GIN	If the terminal is in Alpha mode, performs a carriage return in the dialog area No action if the terminal is in Vector or Marker mode

ENABLE KEY EXPANSION

Enables or disables key macros.

Host Syntax

E_cKW mode

Setup Syntax

KEYEXPAND mode

mode: integer; (keyword in Setup); specifies whether key expansion is enabled. Must be one of the following:

Host Setup

0 no Disables key expansion
1 yes Enables key expansion

Defaults: Factory = 1
Omitted = 1

The ENABLE KEY EXPANSION command enables or disables all key macros.

When key expansion is enabled, the user can expand a key's macro by simply pressing the key. When key expansion is disabled, all keys temporarily revert to their default values.

This command does not delete macros. All key macros remain in program memory and you can reenable them at any time.

This command does not affect how the host uses macros. Even when key expansion is disabled, the host can still issue the EXPAND MACRO command to expand any macro, including those associated with keys.

NOTE

While key expansion is disabled, all programmed key macros revert to their default values.

Related Commands

EXPAND MACRO
DEFINE NONVOLATILE MACRO
LEARN
LEARN NONVOLATILE
MACRO STATUS
SAVE NONVOLATILE PARAMETERS

ENABLE 4010 GIN

Enables the terminal for one 4010 GIN Report.

Host Syntax

E_cSB

When the terminal receives the ENABLE 4010 GIN command, it displays the GIN cursor, which the user positions on the screen by using the Joydisk. Once the cursor is at the desired location, the user reports the location to the host by pressing a key on the keyboard. The terminal then sends the key name and cursor location in a 4010 GIN Report (described in detail in the *Reports* discussion at the end of this section).

After sending the 4010 GIN Report, the terminal updates its graphics position to the GIN cursor location, and enters Alpha mode. This makes the terminal compatible with programs written for Tektronix 4010 Series terminals.

Since the terminal is enabled for only one report, a host program must issue a separate command for each GIN point required.

Emulating the graphics input capability of a Tektronix 4010 Series terminal requires some specific communications settings. While the exact settings may vary from one computer installation to another, the settings shown for the following commands should work for most host computers:

- SET EOM CHARACTERS — Set both EOM characters to N_U .
- SET EOL STRING — Set the EOL string to C_R .
- SET BYPASS CANCEL CHARACTER — Set the bypass cancel character to match to the last character that the host echoes in response to a report's last character. For 4010 GIN, the bypass cancel character will be one of these:
 - N_U if the host is not echoing characters
 - L_F if the host echoes C_R as $C_R L_F$
 - C_R if the host echoes C_R as just C_R

Related Commands

SET BYPASS CANCEL CHARACTER
SET EOL STRING
SET EOM CHARACTERS

END PANEL

Concludes a panel definition.

Host Syntax

```
ⒺcLE
```

Setup Syntax

```
ENDPANEL
```

When you issue this command, the terminal closes the panel boundary, fills the panel with the current fill pattern, and sets the graphics position to the panel boundary's starting point.

See the chart in Appendix F, which shows all of the terminal's predefined fill patterns.

Related Commands

BEGIN PANEL BOUNDARY
SELECT FILL PATTERN

ENQUIRY

Queries the terminal for its answerback string.

Host Syntax

```
ⒺQ
```

The ENQUIRY command invokes the answerback string in all host command modes (ANSI, EDIT, VT52, or TEK mode). The terminal does not respond to this command in Local mode.

Your program can use the answerback string to identify the terminal and determine whether the terminal is authorized to use specific programs and data.

NOTE

The terminal does not enter Bypass mode when it sends the answerback string. If (1) your host echoes characters and (2) you do not want the terminal to echo the answerback string, then you must issue the ENTER BYPASS MODE command before you issue the ENQUIRY command.

If the host provides an echo, it will echo the answerback string. If you don't want the answerback string displayed on the terminal, issue the ENTER BYPASS MODE command before you issue the ENQUIRY command.

The terminal's answerback string can be set by using the Setup command SET ANSWERBACK STRING.

Note that, in TEK mode, the ⒺQ character is a command terminator (like Ⓔc, Ⓔs, Ⓔg, and Ⓔu).

Related Commands

SET ANSWERBACK STRING

ENTER ALPHA MODE

Puts the terminal in Alpha mode.

Host Syntax

```
US
```

The default implicit command mode at power-up is Alpha mode; there are two other implicit command modes: Vector mode and Marker mode.

When the terminal is in Alpha mode, it interprets and displays ASCII characters as alphatext. Alphatext is sent to the dialog buffer if the dialog area is enabled, or to the graphics area if the dialog area is disabled.

If the dialog area is enabled and visible, alphatext is displayed in the dialog area as it is received from the host.

The terminal exits Alpha mode when it receives an ENTER VECTOR MODE or ENTER MARKER MODE command.

Related Commands

ENABLE DIALOG AREA
ENTER MARKER MODE
ENTER VECTOR MODE
LFCR
PAGE
SET ALPHATEXT SIZE
SET ALPHATEXT FONT
SET 4014 ALPHATEXT SIZE

ENTER BYPASS MODE

Puts the terminal in Bypass mode.

Host Syntax

```
ECCN
```

Bypass mode is one of the communications modes — the other communications modes are Prompt mode and Local mode.

When the terminal is in Bypass mode, it ignores all characters from the host until it receives the *bypass cancel character*. When the terminal receives this character, it terminates Bypass mode and discards the bypass cancel character.

The terminal automatically enters Bypass mode when it sends the first character of a report to the host. The bypass cancel character is usually the character that the host sends the terminal after it reads a report from the terminal. Here's a typical sequence:

1. The host requests a report (by issuing a REPORT TERMINAL SETTINGS command).
2. The terminal enters Bypass mode.
3. The terminal sends a report, terminated by the EOL string.
4. The host echoes the report to the terminal, but the terminal ignores it.
5. The host sends the bypass cancel character.
6. The terminal cancels Bypass mode.
7. The host sends more data, which the terminal processes.

The terminal does *not* enter Bypass mode to send reports about ANSI mode or to issue the answerback string. If you want to suppress the echo of the answerback string, you should send the terminal an ENTER BYPASS MODE command before you issue the ENQUIRY command.

If the current bypass cancel character is set to N_U , Bypass mode is disabled; in this case, the ENTER BYPASS MODE command has no effect.

See the discussion on Bypass mode in Section 2 for more information. Also see the SET BYPASS CANCEL CHARACTER command later in this section.

Related Command

SET BYPASS CANCEL CHARACTER

ENTER MARKER MODE

Puts the terminal in Marker mode.

Host Syntax

F_S

Marker mode is one of the three implicit command modes; the other two are Alpha mode and Vector mode.

When the terminal is in Marker mode, it interprets ASCII characters as xy-coordinates and draws markers at the locations specified by the coordinates. (The SET MARKER TYPE command specifies the kind of marker the terminal draws.)

The terminal cannot go directly from Marker mode to Vector mode; it must first be placed in Alpha mode, then in Vector mode.

The discussions *Implicit Command Modes* and *Markers*, both in Section 4, explain how to use Marker mode. Also see the discussion of xy-coordinates under *Host Parameters* (at the beginning of this section) for details on how to encode and send xy-coordinate parameters.

Related Commands

BEGIN PANEL BOUNDARY
DRAW
DRAW MARKER
ENTER ALPHA MODE
ENTER VECTOR MODE
MOVE
SET MARKER TYPE

ENTER VECTOR MODE

Puts the terminal in Vector mode.

Host Syntax

G_S

Vector mode is one of the three implicit command modes; the other two are Alpha mode and Marker mode.

When the terminal is in Vector mode, it interprets ASCII characters as xy-coordinates. The terminal moves the graphics position to the first xy-coordinate, and draws vectors to the subsequent xy-coordinates.

To DRAW rather than MOVE when specifying the first coordinate after entering Vector mode, include the B_L character (ADE 7) immediately after the G_S character.

The discussions *Implicit Command Modes* and *Vectors*, both in Section 4, explain how to use Vector mode. Also see the discussion of xy-coordinates under *Host Parameters* (at the beginning of this section) for details on how to encode and send xy-coordinate parameters.

Related Commands

DRAW
ENTER ALPHA MODE
ENTER MARKER MODE
MOVE
SET LINE INDEX
SET LINE STYLE
SET 4014 LINE STYLE

EXPAND MACRO

Expands a macro.

Host Syntax

$\text{E}c\text{KX}$ macro-number

Setup Syntax

EXPAND macro-number

macro-number: integer; indicates the macro to expand. Valid values range from -150 through 32767 (except -1).

Defaults: Factory = (none)
 Omitted = 0

The EXPAND MACRO command causes the terminal to send the contents of a stored macro definition to the TEK Mode Interpreter (see the discussion on terminal architecture in Section 1). If the contents of the macro make a valid command, the terminal executes it; if the characters are not a valid command, the terminal displays them in the dialog area as alphanext (or in the graphics area if the dialog area is disabled — see the ENABLE DIALOG AREA command).

The macro definition being expanded may contain other EXPAND MACRO commands. You can nest commands to a nesting depth of at least five.

Macros numbered from -150 through 143 may also be expanded by typing the corresponding key on the keyboard.

NOTE

An important difference in expanding a macro from the keyboard is that instead of sending the macro contents to its command interpreter, the terminal sends the macro to the host unless the macro contains key-execute characters. Refer to Keeping a Key Macro Local in Section 4 for details about the key execute-character and how it works.

Syntax Example

Host: $\text{E}c\text{KXH0}$
 Setup: **EXPAND 128**

Expands Macro 128 (128 is encoded *H0*). You could also expand this macro by pressing Function Key F1, which corresponds to Macro Code 128.

Related Commands

DEFINE MACRO
 DEFINE NONVOLATILE MACRO
 LEARN
 LEARN NONVOLATILE
 SET KEY EXECUTE CHARACTER

FACTORY

Sets all parameters to their factory default values and takes the terminal out of Setup.

Setup Syntax

FACTORY

This command restores the terminal to its factory default condition. It erases the contents of the terminal's program memory, including all changes in parameter settings and all macro definitions.

Hint. If you've saved settings in nonvolatile memory and want to return all settings to their factory default, issue the SAVE NONVOLATILE PARAMETERS command (NVSAVE in Setup) after you issue the FACTORY command.

Related Commands

RESET
 SAVE NONVOLATILE PARAMETERS

GRAPHIC TEXT**GRAPHIC TEXT**

Writes a string of graphtext, starting at the current graphics position.

Host Syntax

```
EcLT text
```

Setup Syntax

```
GTEXT text
```

text: string (delimited string in Setup); indicates the characters to be displayed. Each character must be in the range ADE 32 through ADE 126 (Sp through ~).

Defaults: Factory = (none)
Omitted = Empty string

The terminal draws the text string in the direction determined by the SET GRAPHTEXT CHARACTER PATH command. After the string is drawn, the graphics position is updated to a point also determined by the SET GRAPHTEXT CHARACTER PATH command (see the SET GRAPHTEXT CHARACTER PATH command for details).

If the string is too long to fit in terminal space, the terminal clips the characters at the edge of terminal space and sets the current graphics position to the edge of terminal space where the overflow occurred.

You cannot include graphtext in a panel; if you attempt to use the GRAPHIC TEXT command while defining a panel, the terminal detects an error.

The terminal displays graphtext as if it were alphanet, except that it does not wrap at the right edge of the display. The appearance of graphtext is governed by the SET TEXT INDEX, SET GRAPHTEXT SIZE, SET GRAPHTEXT ROTATION, SET ALPHATEXT FONT, and ANSI mode SCS (SELECT CHARACTER SET) commands.

Refer to the discussion *Displaying Text in the Graphics Area* in Section 4.

Syntax Example

```
Host: EcLT7UNICORN
Setup: GTEXT /UNICORN/
```

Writes the string *UNICORN* starting at the current graphics position.

Related Commands

```
SET ALPHATEXT FONT
SET GRAPHTEXT ROTATION
SET GRAPHTEXT SIZE
SET TEXT INDEX
```


HARDCOPY

Copies the contents of the terminal's screen (or just the dialog area) to a copier or printer.

Host Syntax

```
ⒺcKH hardcopy-code
```

hardcopy-code: integer; selects the portion of the display that is copied. Must be one of the following:

- 0 or 1 Copies the entire screen
- 2 Copies the entire screen, reversing black and white
- 3 Copies only the dialog area

Defaults: Factory = (none)
Omitted = 0

The values 0, 1, and 3 create a negative image of the display (white areas copy black, black areas copy white), which is the way a copy normally appears. If you prefer a positive image, use *hardcopy-code 2* in the command.

Using this command has the same effect as pressing the S Copy, Ctrl with S-Copy, or D Copy keys (*hardcopy-codes* 0 or 1, 2, and 3, respectively).

To copy only the graphics area, first make the dialog area invisible (from the host use the SET DIALOG AREA VISIBILITY command, from the keyboard use the Dialog key). Then you can make the screen copy (from the host use the HARDCOPY command with a parameter of 0 or 1, from the keyboard press the S Copy key). After the copy starts, you can make the dialog area visible again and continue working in the dialog area. However, you cannot work in the graphics area until the copy is complete.

On the Tektronix 4691 and 4692 Color Graphics Copiers, dialog copies are always in vertical orientation (the long axis of the image is on the short axis of the paper), and the SET IMAGE ORIENTATION command determines the orientation and size of screen copies.

On the Tektronix 4695 Color Graphics Copier and all monochrome graphics copiers, both screen and dialog copies are in vertical orientation (the long axis of the image is on the short axis of the paper).

On monochrome graphics printers, screen copies will represent the color image in black and white; as a default, all color indices except Index 0 print as black. You can use the MAP INDEX TO PRINT command to control which colors print and which do not print. You can reverse the black and white values by issuing the HARDCOPY command with the *hardcopy-code* parameter set to 2.

If you have selected a monochrome text-only printer with the SELECT HARDCOPY INTERFACE command, requesting a screen hardcopy (S Copy) will generate an error.

Syntax Example

```
Host: ⒺcKH3
```

Sends a copy of the dialog area (*hardcopy-code 3*).

Related Commands

```
COPY
MAP INDEX TO PRINT
SELECT COLOR HARDCOPY IMAGE DENSITY
SELECT HARDCOPY INTERFACE
SET COLOR COPIER REPAINT
SET COPY SIZE
SET DIALOG AREA HARDCOPY ATTRIBUTES
SET HARDCOPY MONOCHROME ATTRIBUTES
SET IMAGE ORIENTATION
4010 HARDCOPY
```

HELP**HELP**

Displays information about a command or cluster of commands.

Setup Syntax

```
HELP name
```

name: string; specifies the command or the name of a cluster of commands for which you want information.

Defaults: Factory = (none)
Omitted = All commands

When you issue this command, the terminal displays (as applicable) the host escape sequence, Setup name, and parameter types (including keywords) for one or more commands.

You can request information on a specific command by issuing either the command's Setup name or its host escape sequence. For example, you can request information on the SET DIALOG AREA LINES command either by typing **HELP DALINES** (the Setup name) or **HELP $\text{E}cLL$** (the escape sequence).

If you enter just a single letter for the *mode* parameter, the terminal displays help information about all commands whose Setup names begin with that letter. If you enter two letters, the terminal displays all commands that begin with those letters. Likewise with three letters, four letters, and so on.

If you enter a cluster name, the terminal displays help information about all commands in that category. The cluster names are *ANSI*, *Communications*, *Dialog*, *General*, *Graphics*, *Hardcopy*, *Keyboard*, and *Pixels*.

Syntax Example

Setup: **HELP dialog**

Queries the terminal to find out what dialog area attributes have been set.

Related Commands

STATUS

IGNORE DELETES

Determines whether the terminal ignores the $\text{D}T$ (Delete) character. (Can be saved in nonvolatile memory.)

Host Syntax

```
 $\text{E}cKI$  ignore-deletes-mode
```

Setup Syntax

```
IGNOREDEL ignore-deletes-mode
```

ignore-deletes-mode: integer (keyword in Setup); specifies whether the terminal ignores $\text{D}T$ characters. Must be one of the following:

Host	Setup	Description
0	no	Doesn't ignore $\text{D}T$ characters
1	yes	Ignores $\text{D}T$ characters

Defaults: Factory = 0
Omitted = 1

Some computers use $\text{D}T$ characters (Delete or Rubout — ADE 127) as filler characters, sprinkling them throughout the data stream sent to the terminal. These, in effect, slow down the communications rate and give the terminal time to react to information from the host without losing data or having to reset the baud rate.

Since $\text{D}T$ is a valid character in integer and xy-coordinate parameters, there are times when the terminal should process $\text{D}T$ characters as data, and other times when it should treat the character as a filler character and ignore it. This command tells the terminal which way to treat the $\text{D}T$ character.

See *$\text{D}T$ Filler Characters* in Section 2 for details on how to use this command.

LEARN

Programs a key from the keyboard.

Setup Syntax

LEARN

A key programmed with the LEARN command remains programmed only until the terminal is turned off or reset. To save a key macro, you must use the LEARN NONVOLATILE command.

To program a key, first enter Setup, then type:

LEARN

The terminal will respond with the message:

Press the key to be defined:

The key you press can be any key on the keyboard (except Caps Lock, Ctrl, and Shift) including function keys, the Break key, a shifted space bar, etc. If you press an alphanumeric key, the terminal echoes the corresponding ASCII character on the screen. If you press a function key or a key combination, such as pressing Shift and Return simultaneously, the terminal echoes the decimal macro number.

Now simply type in the definition (you don't need delimiters). If you make a mistake, use F2 to delete characters. The terminal adds the ASCII character for each key you press to the definition. When you press F1 the terminal ends the definition and puts you back into Setup. If you press any key (except F1 and F2) that lacks an ASCII definition while typing the definition, the terminal ignores that keystroke and rings its bell.

You can cancel the definition before ending it by pressing the Cancel key.

If there is not enough program memory available to store a definition when you issue the LEARN command, the terminal returns to Setup after displaying this message:

Error: Not enough memory available.

If, however, program memory fills up *while* you are writing the definition, the terminal rings its bell and ignores any other keys you press except the F1 key or the Cancel key. Pressing the F1 function key will unlock the keyboard, terminate the definition, and program the key with as much of the definition as you have completed. Pressing the Cancel key will unlock the keyboard, but the definition you have entered will be lost.

Programming keys uses program memory that could be used for other features such as the input queue or panel definitions. See the discussion *Managing Program Memory* in Section 4 for an explanation of how programmed keys affect the availability of program memory.

Related Commands

DEFINE MACRO
EXPAND MACRO
LEARN NONVOLATILE

LEARN NONVOLATILE**LEARN NONVOLATILE**

Programs a key from the keyboard so that the definition can be saved in nonvolatile memory.

Setup Syntax

```
NVLEARN
```

A key programmed with the LEARN NONVOLATILE command *and* saved with the SAVE NONVOLATILE PARAMETERS command remains programmed until you reprogram it, even if the terminal is turned off.

NOTE

Key definitions programmed with the LEARN NONVOLATILE command are saved in nonvolatile memory only if you issue a SAVE NONVOLATILE PARAMETERS command before you reset or turn off the terminal or issue the FACTORY or RESET command.

To program a key, first enter Setup, then type:

```
NVLEARN
```

The rest of the key programming procedure is exactly the same as for the LEARN command. Refer to the LEARN command description for details.

Related Commands

```
DEFINE NONVOLATILE MACRO
EXPAND MACRO
LEARN
```

LFCR

Specifies whether a LF character sent to the terminal also implies a CR . (Can be saved in nonvolatile memory.)

Host Syntax

```
 $\text{E}_c\text{KF}$  lfcr-mode
```

Setup Syntax

```
LFCR lfcr-mode
```

lfcr-mode: integer (keyword in Setup); must be one of the following:

Host	Setup	
0	no	LF does not imply CR
1	yes	LF implies CR

Defaults: Factory = 0
Omitted = 1

When LF does not imply CR , the LF character is displayed as a Line Feed only, not as a Line Feed and Carriage Return combination.

When LF implies CR , a LF character is displayed as a Line Feed and Carriage Return combination.

This setting affects only a LF sent to the terminal screen. When you press the Line Feed key, for example, the implied CR character is not sent to the host.

The ANSI commands SM (SET MODE) and RM (RESET MODE) can perform the same action as the LFCR command by setting and resetting Linefeed/Newline mode (LNM).

Related Commands

```
CRLF
LNM (LINEFEED/NEWLINE MODE)1
```

¹ This is an ANSI command, described in Section 3.

LOCAL

Specifies whether the terminal processes commands from the host or is controlled from its own keyboard.

Setup Syntax

```
LOCAL local-mode
```

local-mode: keyword: specifies whether to enter or exit Local mode. Must be one of the following:

yes Initiates Local mode
no Cancels Local mode

Defaults: Factory = no
Omitted = yes

When the terminal is in Local mode, it responds to command escape-sequences typed on the keyboard as though they came from the host. Other characters typed on the keyboard are displayed on the screen.

In Local mode, the terminal does not respond to characters sent from the host. Instead, the terminal stores them in its communications input queue.

Once you issue this command, you must press the Setup key to leave Setup in order to actually enter Local mode.

The terminal does not respond to the ENQUIRY command in Local mode.

LOCK KEYBOARD

Locks or unlocks the keyboard.

Host Syntax

```
EcKL locking-mode
```

locking-mode: integer; specifies whether the keyboard is locked or unlocked. Must be one of the following:

0 Unlocks the keyboard
1 Locks the keyboard

Defaults: Factory = 0
Omitted = 0

The LOCK KEYBOARD command lets the host computer disable the keyboard keys — this is useful at times when a host computer program cannot tolerate input from the user. When the keyboard is locked, the Joydisk and all the keys except Cancel and Break are inoperative. Pressing any key on the keyboard or manipulating the Joydisk will ring the bell.

The user can tell if the keyboard is locked by pressing the Shift key. If the keyboard is locked, the bell will ring. If it is unlocked, no character will be transmitted and the terminal will not sound its bell.

Other ways to unlock the keyboard (besides issuing this command) are pressing the Cancel or Break key, or issuing a CANCEL command.

Related Commands

CANCEL
EMI (ENABLE MANUAL INPUT)¹
DMI (DISABLE MANUAL INPUT)¹

¹ These are ANSI commands, described in Section 3.

MACRO STATUS**MACRO STATUS**

Displays a macro definition.

Setup Syntax

```
MACROSTATUS macro-number
```

macro-number: integer; specifies which macro definition you want displayed. Must be in the range -150 through 32767. Specifying -1 or the keyword *all* displays all macros.

Default: Factory = (none)
Omitted = 0

When you issue this command, the terminal displays the macro's number and definition. (Refer to Appendix A for the macro numbers associated with keys.)

Related Commands

```
DEFINE MACRO
DEFINE NONVOLATILE MACRO
EXPAND MACRO
LEARN
```

MAP INDEX TO PRINT

Specifies which graphics color indices print and which do not print on monochrome printers. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcQI monochrome-values
```

Setup Syntax

```
HCMAP monochrome-values
```

monochrome-values: integer array; each pair of integers specifies an index number (-1 — 7) and a print value (Index -1 specifies all indices). Valid print values are:

```
1 Print
0 Don't print
```

Defaults: Factory = 0 (no print) for Index 0
1 (print) for Indices 1—7
Omitted = Error

Some graphics images created in color won't be readable in monochrome, since all indices will print black (except Index 0). This command allows you to select which indices print black and which don't print at all. By using this command, you can make monochrome copies that offer all the information you intended.

Use the MAP INDEX TO PRINT command to tell the terminal which graphics color indices to print. Since the default is to print all indices except the background index (Index 0), you will most often use this command to suppress printing of some indices.

Dialog area indices are not affected by this command. Text displayed in the dialog area will print, unless you make the dialog area invisible by pressing the Dialog key.

Each pair in the *monochrome-values* array consists of an index number and a print value:

- The index number must be a value between -1 and 7. If you specify -1 here, you can define a single print value for all graphics color indices.
- The print value specifies which color indices will print in monochrome copies. A color index assigned a print value of 1 will print black; an index with a print value of 0 will not print.

Syntax Example

Host: `EcQI42040`
 Setup: `HCMAP 2,0,4,0`

Suppresses printing of Indices 2 and 4, allowing all other indices (except Index 0) to print. In the host example, the 4 following the opcode is an array count indicating that there are four integers in the array.

Related Commands

COPY
 HARDCOPY

MOVE

Moves the graphics position without drawing a vector.

Host Syntax

`EcLF position`

Setup Syntax

`MOVE position`

position: xy-coordinate; specifies the new graphics position. Valid range of values is 0 through 4095 for both the x- and y-coordinate.

Defaults: Factory = (none)
 Omitted = 0,0

The MOVE command's operation is analogous to lifting a pen from the paper in a drawing and moving it to a new location. Unless you are defining a panel boundary, this command does not draw anything visible; it simply moves the graphics position to a new starting point for other graphics commands, such as DRAW and GRAPHIC TEXT.

The discussion titled *Vectors* in Section 4 gives examples of how to use this command with the DRAW command to create lines.

Syntax Example

Host: `EcLF \ azSPM`
 Setup: `MOVE 53,1000`

Moves the current graphics position to location 53,1000 without drawing a vector (the xy-coordinate 53,1000 is encoded `\ azSPM`).

Related Commands

BEGIN PANEL
 DRAW
 ENTER VECTOR MODE
 GRAPHIC TEXT
 SET LINE STYLE
 SET 4014 LINE STYLE

PAGE**PAGE**

Erases the graphics area.

Host Syntax

```
EcFf
```

This command has the same effect as pressing the terminal's G Eras key.

If the dialog area is enabled, the terminal erases the graphics area.

If the dialog area is not enabled, the terminal does the following:

- Erases the graphics area
- Resets the current line style to 0 (solid lines)
- Terminates 4010 GIN mode (if it was enabled)
- Sets the current graphics position to the home position
- Enters Alpha mode

Related Commands

ENABLE DIALOG AREA

PIXEL COPY

Copies pixels from one rectangular region to another.

Host Syntax

```
EcRX destination-surface
destination-lower-left-corner
first-source-corner
second-source-corner
```

Setup Syntax

```
PXCOPY destination-surface
destination-lower-left-corner
first-source-corner
second-source-corner
```

destination-surface: integer; names the surface to which pixels are to be copied. The values -1, 0, and 1 are valid, but since Surface 1 is the terminal's only surface, all valid values select Surface 1. (Other Tektronix terminals can display more than one surface, so the surface parameter is included here for compatibility.)

Defaults: Factory = (none)
Omitted = 0

destination-lower-left-corner: xy-coordinate; specifies the lower-left corner of the rectangular region that will receive the copy. Valid range of values for x is 0 through 479; for y, 0 through 359.

Defaults: Factory = (none)
Omitted = 0,0

first-source-corner: xy-coordinate; specifies any corner of the rectangular region that you want to copy. Valid range of values for x is 0 through 479; for y, 0 through 359.

Defaults: Factory = (none)
Omitted = 0,0

second-source-corner: xy-coordinate; specifies the corner opposite the *first-source-corner*. Valid range of values for x is 0 through 479; for y, 0 through 359.

Defaults: Factory = (none)
Omitted = 0,0

This command copies pixels from one region to another. It copies the pixel at the *first-source-corner* to the lower-left corner of the destination region, which is the same width and height as the source region. Then it copies each remaining pixel in the source region onto a corresponding pixel in the destination region.

The two source corners need not be the lower-left and upper-right corners of the source region. If they aren't, however, the pixels written to the destination region may form a mirror (or inverted) image of the picture in the source region.

You can create some special effects by copying pixels to the same location, depending on the ALU mode specified in the BEGIN PIXEL OPERATIONS command. If the ALU mode is set to XOR, you can erase pixels by copying them to the same location. In other ALU modes, you can create a mirror image by specifying the upper-right source corner first. Using ALU modes other than XOR to copy (without mirroring) to the same location will do nothing.

You can copy pixels to the off-screen graphics memory, which has x values from 480 to 511, but a Level 0 warning will be generated.

Syntax Example

Host: `EcRX1"pk"K!pb!B!zt!T`
 Setup: `PXCOPY 1,300,300,200,200,210,210`

Copies an area that has one corner at 200,200 and another corner at 210,210 to an area whose lower-left corner is at 300,300. In the host example, the xy-coordinate pair 300,300 is encoded "*pk*"*K*; the xy-coordinate pair 200,200 is encoded *!pb!B*; the xy-coordinate pair 210,210 is encoded *!zt!T*.

Related Command

BEGIN PIXEL OPERATIONS

PROMPT MODE

Turns Prompt mode on or off.

Host Syntax

`EcNM prompt-mode`

Setup Syntax

`PROMPTMODE prompt-mode`

prompt-mode: integer (keyword in Setup); must be one of the following:

<u>Host</u>	<u>Setup</u>	
0	no	Cancels Prompt mode
1	yes	Initiates Prompt mode after the next EOM character or EOL string
2	—	Initiates Prompt mode immediately (host syntax only)

Defaults: Factory = 0
 Omitted = 1

See Section 2 for an explanation of how Prompt mode works.

Related Commands

- SET EOL STRING
- SET EOM CHARACTERS
- SET PROMPT STRING
- SET TRANSMIT DELAY

RASTER WRITE**RASTER WRITE**

Sets the color indices individually for one or more pixels in the pixel viewport.

Host Syntax

```

EcRP number-of-pixels
      color-index-codes
  
```

Setup Syntax

```

PXRASTERWRITE number-of-pixels
              color-index-codes
  
```

number-of-pixels: integer; specifies how many pixels are to receive a color index. Must be in the range 0 through 65535.

Defaults: Factory = (none)
Omitted = Error

color-index-codes: string (delimited string in Setup); specifies the color indices for the pixels specified by *number-of-pixels*. Must be in a special format (described later in this discussion) which uses characters whose ADE must be in the range 32 through 96 (S_P through \).

Defaults: Factory = (none)
Omitted = 0

This command sets the color for each pixel in a string of pixels, starting at the current pixel beam position in the pixel viewport.

Figure 5-6 shows how to pack color indices into the *color-index-codes* parameter. To see how the format for the *color-index-codes* parameter works, convert the characters in the string to their binary equivalents. Picture this data string as a sequence of bits — to form color indices for individual pixels, the terminal groups these bits using the value of the *bits-per-pixel* parameter in the BEGIN PIXEL OPERATIONS command.

Also refer to the pseudocode routine for sending the RASTER WRITE command under *Writing Into the Pixel Viewport* in Section 4.

If *bits-per-pixel* in the BEGIN PIXEL OPERATIONS command is set to 1, then six color indices (each consisting of a single bit) will fit into each code character. If *bits-per-pixel* is 2, then three color indices fit into each code character. Figure 5-6 shows how two color indices fit into each code character if *bits-per-pixel* is 3.

If *bits-per-pixel* in the BEGIN PIXEL OPERATIONS command is 4 or 6, the terminal interprets each code character as containing only one color index, which is determined by the four least significant bits. You can represent each color index in the range 0 through 7 with a single ASCII character.

The special character \ (ADE 96) functions much like a C_RT_F sequence; it moves the pixel beam position to the start of the following row of pixels. The \ code is not included in the pixel count.

Syntax Example

```

Host:   EcRP99222333222
Setup:  PXRASTERWRITE 9,/222333222/
  
```

Assuming *bits-per-pixel* has been set to 6 in the last BEGIN PIXEL OPERATIONS command, this example specifies indices for nine pixels, setting the first three pixels to Index 2, the next three pixels to Index 3, and the last three pixels to Index 2.

Related Commands

```

BEGIN PIXEL OPERATIONS
RUNLENGTH WRITE
SET PIXEL BEAM POSITION
SET PIXEL VIEWPORT
  
```

If *bits-per-pixel* is 3, then pack the color indices 0, 0, 2, 3, 2, 7 into a RASTER WRITE command as follows:

- Express the color indices as three-bit binary numerals:

0	0	2	3	2	7
↓	↓	↓	↓	↓	↓
000	000	010	011	010	111

- Group the binary bits into six-bit groups:

000 000	010 011	010 111
↓	↓	↓
000000	010011	010111

- Add 32 (binary 100000) to these six-bit binary numerals to form seven-bit ASCII characters:

0100000	0110011	0110111
↓	↓	↓
S _P	3	7

- Issue a RASTER WRITE command. The command's first parameter is the integer 6, because the command holds six color indices. The second parameter is a character array holding the characters S_P, 3, and 7.

RASTER WRITE = E_cRP 6 3_{S_P}37

4526-39

Figure 5-6. Packing Color Indices Using Three Bits per Pixel.

RECTANGLE FILL**RECTANGLE FILL**

Sets all the pixels in a rectangle to the same color.

Host Syntax

```
EcRR lower-left-corner
      upper-right-corner
      fill-index
```

Setup Syntax

```
PXRECTANGLE lower-left-corner
             upper-right-corner
             fill-index
```

lower-left-corner: xy-coordinate; specifies one corner of a rectangle in graphics memory space. Valid range of values for x is 0 through 479; for y, 0 through 359.

Defaults: Factory = (none)
Omitted = 0,0

upper-right-corner: xy-coordinate; specifies the opposite corner of that rectangle. Valid range of values for x is 0 through 479; for y, 0 through 359.

Defaults: Factory = (none)
Omitted = 0,0

fill-index: integer; specifies the color index used to fill the rectangle. Must be in the range 0 to 65535.

Defaults: Factory = (none)
Omitted = 0

The terminal writes color indices into graphics memory using the ALU mode specified in the BEGIN PIXEL OPERATIONS command.

If the lower-left and upper-right corners of the rectangle have the same x value, then the rectangle filled is one pixel wide. Likewise, if the lower-left and upper-right y values are the same, then the rectangle filled is one pixel high.

This command also functions in off-screen graphics memory where the x value is 480 through 511; however, a Level 0 warning will be generated.

Syntax Example

```
Host: EcRR$ppy$P$Y"$Dty#W3
Setup: PXRECTANGLE 100,100,479,300,3
```

Sets all the pixels in the rectangle with corners 100,100 and 479,359 to Index 3. In the host example, the xy-coordinate 100,100 has been encoded \$ppy\$P\$Y; the xy-coordinate 479,379 has been encoded "\$Dty#W.

Related Commands

```
BEGIN PIXEL OPERATIONS
RASTER WRITE
RUNLENGTH WRITE
```

REPORT ERRORS

Queries the terminal for an Error Report listing the eight most recent errors.

Host Syntax

```
EcKQ
```

When the host requests an Error Report, the terminal reports the eight most recently detected error codes, their severity levels, and how many times each error was detected.

If fewer than eight errors have been detected since power-up or since the last REPORT ERRORS command, then the terminal sends fewer than eight *reports-for-one-error*.

For details, see the Error Report description in the *Reports* discussion at the end of this section.

Related Commands

```
ENTER BYPASS MODE
```

REPORT SYNTAX MODE

Queries the terminal for a Terminal Settings Report giving the current host command mode (ANSI, EDIT, TEK, or VT52).

Host Syntax

`Ec#10`

This command has the same effect as a REPORT TERMINAL SETTINGS command issued for the SELECT CODE command (as if the host sent `EcIQ%!`). See the REPORT TERMINAL SETTINGS command.

This command is recognized in all host command modes.

You can display the host command mode status on the screen by entering the Setup command *STATUS CODE*.

Related Commands

REPORT TERMINAL SETTINGS
 SELECT CODE

REPORT TERMINAL SETTINGS

Queries the terminal for a Terminal Settings Report.

Host Syntax

`EcIQ inquiry-code`

inquiry-code: two characters; specifies the two-letter opcode for an escape-sequence command or a special two-character inquiry code for other information about the terminal.

Defaults: Factory = (none)
 Omitted = 0 (no effect)

This general-purpose inquiry command tells the terminal to send a Terminal Settings Report to the host (see the *Reports* discussion at the end of this section).

Besides the opcodes for commands, you can also use *special inquiry codes* — Table 5-3 lists the three special inquiry codes and the information they query for.

You can query for terminal settings status from the terminal keyboard by issuing the STATUS command in Setup — the settings will be displayed on the terminal screen. The STATUS command uses keywords to make the special inquiry requests described in Table 5-3.

Related Commands

REPORT SYNTAX MODE
 STATUS

Table 5-3
SPECIAL INQUIRY CODES

Code	Report Contents
?M	Two integers report (1) available program memory and (2) the largest contiguous block of program memory (both reported as a number of 16-byte units of memory). In Setup enter <i>STATUS MEMORYBLOCKS</i> . ^a
?T	An integer reports the terminal model number. In Setup enter <i>STATUS TERMINAL</i> . ^a
00	An integer reports the firmware version installed in the terminal. In Setup enter <i>STATUS VERSION</i> . ^a

^a The Setup command STATUS does not send a report but displays the equivalent information on the terminal screen.

REPORT 4010 STATUS

Queries the terminal for a 4010 Status Report, which gives the alpha cursor position and copier status (or just GIN cursor position if 4010 GIN is enabled).

Host Syntax

```
ECEQ
```

This command also terminates 4010 GIN and puts the terminal in Alpha mode.

See the description of the 4010 Status Report in the *Reports* discussion at the end of this section.

Related Commands

ENABLE 4010 GIN
SET BYPASS CANCEL CHARACTER

RESET

Returns the terminal to its *power-up condition*.

Host Syntax

```
ECKV
```

Setup Syntax

```
RESET
```

CAUTION

If any of the terminal's current settings for communications parameters differ from settings saved in nonvolatile memory, issuing a RESET command can disrupt host/terminal communications.

The RESET command initializes the terminal to its *power-up condition* (a combination of factory default settings and any settings that have been saved in nonvolatile memory). It is equivalent to pressing the terminal's RESET button or turning the terminal off and then turning it on again.

The terminal takes several seconds to execute the RESET command. During that time, it is performing its power-up reset and self-test routines, and cannot process data coming from the host.

RUNLENGTH WRITE

Sets one or more pixels in the pixel viewport to the same color.

Host Syntax

```
EcRL runcode-array
```

Setup Syntax

```
PXRUNLENGTHWRITE runcode-array
```

runcode-array: integer array; assigns color indices to a specified number of pixels in the pixel viewport. Each runcode in the array can range from 0 through 65535.

Defaults: Factory = (none)

Omitted = Empty array

This command writes color indices into graphics memory using the ALU mode specified in the BEGIN PIXEL OPERATIONS command.

Starting at the current pixel beam position in the pixel viewport, the terminal sets the specified number of pixels to the specified color index for each runcode in the *runcode-array* parameter. When all the pixels for a given runcode have been loaded with the specified color index, the process is repeated for the next runcode in the integer array.

For each runcode, the pixel beam moves to the right, pixel-by-pixel, assigning a color index to each. Upon reaching the right edge of the pixel viewport, the pixel beam jumps back to the left edge on the next line down (or the top line if the beam is at the bottom line's right edge) and continues assigning color indices to pixels until the runcode specification is met.

Each runcode includes two numbers packed together: one is a color index, and the other is the number of pixels that are to be set to that color index. The runcodes are packed using the form:

$$\text{Runcode} = \text{number-of-pixels} * 2^n + \text{color-index}$$

where n = bits-per-pixel

The *bits-per-pixel* parameter from the most recent BEGIN PIXEL OPERATIONS command supplies the value for n , unless that parameter is 4 or 6; then the value of n is 3. Section 4 shows a pseudocode routine for packing runcodes (in the discussion of the RUNLENGTH WRITE command under *Writing Into the Pixel Viewport*).

Syntax Example

Host: **E**_c**RL****I****E****4**

Setup: **PXRUNLENGTHWRITE 84**

Assuming that *bits-per-pixel* is set to 4, this example sets five pixels to Index 4 ($5 * (2^4) + 4 = 84$). In the host example, the *I* following the escape sequence *RL* is the array count; 84 is encoded *E4*.

Related Commands

BEGIN FILL PATTERN
 BEGIN PIXEL OPERATIONS
 RASTER WRITE

SAVE NONVOLATILE PARAMETERS

Saves the values of those commands whose settings can be saved in nonvolatile memory; also saves all nonvolatile macros.

Host Syntax

```
EcKU
```

Setup Syntax

```
NVSAVE
```

This command allows you to save the effects of some commands so that the terminal always powers up with the settings you need for your application. The terminal's power-up condition is a combination of the factory default values and the settings you save in nonvolatile memory.

When you issue **SAVE NONVOLATILE PARAMETERS**, the terminal writes to its nonvolatile memory all settings that have been changed and all nonvolatile macros that have been defined since the last **SAVE NONVOLATILE PARAMETERS** command was issued. The terminal retains those settings and they become part of the terminal's power-up condition.

This command saves only those settings that have changed since the last time this command was issued. The only macros that it saves are those defined with the **DEFINE NONVOLATILE MACRO** and **LEARN NONVOLATILE** commands.

You can identify those commands that set values you can save by referring to their purpose statements at the top of the command description. Look for the statement *can be saved in nonvolatile memory*.

NOTE

*The **SAVE NONVOLATILE PARAMETERS** command may take up to several seconds to complete, depending on the number of changed settings and nonvolatile macros that require saving.*

Each byte of nonvolatile memory has a lifetime of about 10,000 writes. If you attempt to write to nonvolatile memory after that, the terminal might display an error message (depending on the **SET ERROR THRESHOLD LEVEL** setting). When that occurs, all settings are reset to factory default the next time the terminal is powered up, and the terminal can no longer use its nonvolatile memory. (You can have the nonvolatile memory replaced; see your local Tektronix Field Office.)

Related Commands

DEFINE NONVOLATILE MACRO
LEARN NONVOLATILE

SELECT CODE

Causes the terminal to recognize the syntax of host commands in ANSI, EDIT, VT52, or TEK mode. (Can be saved in nonvolatile memory.)

Host Syntax

```
Ec%! syntax
```

Setup Syntax

```
CODE syntax
```

syntax: integer; (keyword in Setup); selects one of the following syntax modes:

Host	Setup	
0	TEK	TEK mode syntax
1	ANSI	ANSI mode syntax
2	EDIT	ANSI mode syntax for EDIT mode
3	VT52	VT52 mode syntax

Defaults: Factory = 0
 Omitted = 0

This command causes the terminal to accept ANSI, TEK, or VT52 commands from the host computer. It does not affect the availability of commands entered from the keyboard in Setup.

The syntax of TEK, ANSI, and VT52 mode commands are not compatible. If you are using commands from one mode and want to execute commands from another mode, you must issue the SELECT CODE command with the appropriate parameter.

This command is recognized in all major modes: ANSI, EDIT, TEK, and VT52.

You can use this command while in Setup to cause the terminal to recognize a specific command syntax when it leaves Setup.

In TEK mode, the terminal:

- Recognizes 4100-style commands from the host

In ANSI mode, the terminal:

- Recognizes ANSI mode commands from the host

In EDIT mode, the terminal:

- Recognizes ANSI mode commands from the host
- Configures the terminal to run VT100 applications:
 - Sets Origin mode to Absolute
 - Sets dialog area and dialog area buffer to 24 lines
 - Makes the dialog area visible
 - Defines a scrolling region of 24 lines
 - Sets Insert/Replace mode to Replace
 - Disables all programmed keys

In VT52 mode, the terminal:

- Recognizes VT52-style commands from the host

NOTE

When switching from EDIT mode to TEK mode, you may need to reset some of your TEK mode settings.

EDIT Mode Settings. EDIT mode sets the terminal's display and keyboard characteristics to emulate a VT100 terminal. When your program returns to TEK mode from EDIT mode, you may need to reset some of the terminal control settings. Specifically:

- EDIT mode disables key expansion; if your program uses programmed keys, issue the ENABLE KEY EXPANSION command when you return to TEK mode.
- EDIT mode sets the dialog area and dialog area buffer to 24 lines; issue the SET DIALOG AREA BUFFER and SET DIALOG AREA LINES commands when you return to TEK mode to set the values you want for your application.
- When you select TEK mode, new dialog overwrites existing data in the dialog area rather than adding it to the end and scrolling — and the combination of old and new information in the dialog area may be confusing. To avoid this, issue the CLEAR DIALOG AREA command after returning to TEK mode.

Related Commands

CLEAR DIALOG AREA
 ENABLE KEY EXPANSION
 REPORT SYNTAX MODE
 REPORT TERMINAL SETTINGS
 SET DIALOG AREA BUFFER
 SET DIALOG AREA LINES

SELECT COLOR HARDCOPY IMAGE DENSITY**SELECT COLOR HARDCOPY IMAGE DENSITY**

Selects the number of dots per inch for color copies. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcQU density-code
```

Setup Syntax

```
HCDENSITY density-code
```

density-code: integer (keyword in Setup); specifies the image density. Must be one of the following:

Host	Setup	
0	low	Low density
1	high	High density

Defaults: Factory = 1
Omitted = 1

The *density-code* selects the number of color dots per inch to print on the copy paper. On the Tektronix 4692 Color Graphics Copier, low density is 144 dots-per-inch, and high density is 155 dots-per-inch. The image size for high density copies is slightly smaller because of the more closely spaced color dots.

This command only affects copies made on the 4692 Copier.

Related Commands

COPY
HARDCOPY
SELECT HARDCOPY INTERFACE
SET COLOR COPIER REPAINT

SELECT FILL PATTERN

Selects a color or predefined pattern to fill a panel.

Host Syntax

```
EcMP fill-pattern-number
```

Setup Syntax

```
FILLPATTERN fill-pattern-number
```

fill-pattern-number: integer; specifies a panel's fill pattern.

Valid values are:

-7 — 0	Solid colors
1 — 16	Predefined patterns
50 — 174	Dithered patterns

Defaults: Factory = -1
Omitted = 0

Notice that the solid colors are indicated by the negative value of a color index (for example, -3 means "fill the panel with Index 3").

Appendix F shows all the fill patterns and their associated fill pattern numbers.

Syntax Example

```
Host: EcMPA0
Setup: FILLPATTERN 16
```

Specifies Fill Pattern 16 (encoded A0) for subsequent panels.

Related Commands

BEGIN PANEL BOUNDARY
END PANEL

SELECT HARDCOPY INTERFACE

Selects the type of copier or printer to be used in making copies. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcQD copier-type
```

Setup Syntax

```
HCINTERFACE copier-type
```

copier-type: integer; identifies the type of copier connected to the terminal. Must be one of the following:

- 0 A monochrome printer with a Centronics-type parallel interface; for copying dialog only
- 1 or 2 A Tektronix 4691, 4692, or 4695 Color Graphics Copier
- 3 A Tektronix 4644 Dot Matrix Printer or other printer with a Centronics-type parallel interface and Epson FX-80 graphics protocol
- 4 A Hewlett-Packard ThinkJet Printer

Defaults: Factory = 2
 Omitted = 0

You must use this command to tell the terminal what kind of copier you're using.

You can make color graphics copies on Tektronix 4691, 4692, or 4695 Color Graphics Copiers. Monochrome graphics copies (which are faster) can be made on a Tektronix 4644 Dot Matrix Printer, a Hewlett-Packard ThinkJet, or other monochrome printers with Epson FX-80 graphics.

You can make dialog copies on some black-and-white printers that use a Centronics-style parallel interface.

Syntax Example

```
Host: EcQD2
Setup: HCINTERFACE 2
```

Tells the terminal that you are using a 4695 Color Graphics Copier (*copier-type* 2).

Related Commands

```
COPY
HARDCOPY
4010 HARDCOPY
```

SET ALPHA CURSOR INDICES

Assigns color indices to the alpha cursor. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcTD first-index
second-index
```

Setup Syntax

```
ACURSOR first-index
second-index
```

first-index: integer; specifies the first color for the alpha cursor; must be in the range 0 to 65535. There are eight color indices available, corresponding to values 0 through 7; a value greater than 7 sets *first-index* to 7.

Defaults: Factory = 1
 Omitted = 0

second-index: integer; specifies the second color for the alpha cursor; must be in the range 0 to 65535. There are eight color indices available on the terminal, corresponding to values 0 through 7; a value greater than 7 sets *second-index* to 7.

Defaults: Factory = 0
 Omitted = 0

The alpha cursor appears on the screen where the next alphanumeric character will be displayed. If *second-index* is a different color than *first-index*, the cursor blinks between the two colors. If the two indices are the same, the cursor does not blink.

The dialog area and the graphics area each have their own set of color indices. When the dialog area is enabled, the alpha cursor indices refer to dialog area indices. When the dialog area is disabled, the alpha cursor indices refer to graphics area indices.

Syntax Example

```
Host: EcTD23
Setup: ACURSOR 2,3
```

Assigns Color Indices 2 and 3 to the alpha cursor so that the cursor blinks (alternating between Index 2 and Index 3).

Related Commands

```
SET DIALOG AREA COLOR MAP
```

SET ALPHATEXT FONT

Selects the font to be used for alphatext and graphtext.

Host Syntax

```
^c font-code
```

font-code: character; selects the G0 or G1 character set.

Must be one of the following:

- s₁ The G0 character set
- s₀ The G1 character set

Defaults: Factory = G0 character set
Omitted = None

The character sets that are the current G0 and G1 character sets can be selected with the ANSI command SCS (SELECT CHARACTER SET).

The SET ALPHATEXT FONT command does not control the character set displayed in Setup.

Related Commands

BEGIN SEGMENT
GRAPHIC TEXT

SET ANSWERBACK STRING

Assigns the terminal's answerback string. (Can be saved in nonvolatile memory.)

Setup Syntax

```
ANSWERBACK answerback-string
```

answerback-string: delimited string; specifies up to twenty characters as the answerback string.

Defaults: Factory = Empty string
Omitted = Empty string

The answerback string acts like a password known only to the terminal and the host application. When the host sends an ENQUIRY command, the terminal responds by transmitting its answerback string. (You can send the answerback string from the keyboard by pressing Ctrl-Break.)

The host can verify the answerback string against a list of authorized users and thus control the data and programs that it lets the terminal access.

NOTE

The string you set with this command is not saved in nonvolatile memory until you issue the SAVE NONVOLATILE PARAMETERS command.

Issuing the SET ANSWERBACK STRING command does not save the answerback string in nonvolatile memory; you must issue a separate SAVE NONVOLATILE PARAMETERS command for the terminal to retain the answerback string in nonvolatile memory.

The terminal transmits only the characters actually in the answerback string. If you want the answerback string to end with a C_R, you must program a C_R into the string when you define it.

Syntax Example

```
Setup: ANSWERBACK /PASSKEY/  
NVSAVE
```

Assigns the string *PASSKEY* as the answerback string.

Related Command

ENQUIRY

SET BAUD RATES

Sets the terminal's transmit and receive baud rates. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcNR transmit-data-rate
receive-data-rate
```

Setup Syntax

```
BAUDRATE transmit-data-rate
receive-data-rate
```

transmit-data-rate: integer; specifies the baud rate at which the terminal sends data to the host. Valid values are 1 (which means *external clock*), 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600, 19200, and 38400 bits per second.

Defaults: Factory = 2400
Omitted = Error

receive-data-rate: integer; specifies the baud rate at which the terminal expects to receive data from the host. Valid values are the same as for *transmit-data-rate* with the addition of 0, which means *same as the transmit rate*.

Defaults: Factory = 2400
Omitted = Same as *transmit-data-rate*

This command sets internally controlled transmit and receive baud rates from 75 to 38400 bits per second or allows transmission rates to be controlled by an external clock.

The transmit and receive parameters need not be the same, unless you set the baud rate to 38400.

Syntax Example

```
Host: EcNRe8R<
Setup: BAUDRATE 600,300
```

Sets a transmit baud rate of 600 (encoded *e8*) and a receive baud rate of 300 (encoded *R<*).

Related Commands

```
PROMPT MODE
SET FLAGGING MODE
SET TRANSMIT RATE LIMIT
SET TRANSMIT DELAY
SET PORT BAUD RATE
SET QUEUE SIZE
```

SET BREAK TIME

Sets the duration (in milliseconds) of the terminal's break signal. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcNK break-time
```

Setup Syntax

```
BREAKTIME break-time
```

break-time: integer; specifies the length of the break signal (in milliseconds). Must be in the range 0 through 65535; a value of 0 disables the break signal.

Defaults: Factory = 200
Omitted = 0

Syntax Example

```
Host: EcNKA9
Setup: BREAKTIME 25
```

Specifies a 25-millisecond break signal when the Break key is pressed (25 is encoded *A9*).

SET BYPASS CANCEL CHARACTER

Specifies the character that cancels Bypass mode. (Can be saved in nonvolatile memory.)

Host Syntax

```
^cNU bypass-cancel-character
```

Setup Syntax

```
BYPASSCANCEL bypass-cancel-character
```

bypass-cancel-character: integer (small integer in Setup); specifies the ADE of the character that cancels Bypass mode. Must be in the range 0 through 127.

Defaults: Factory = 10 (LF)
Omitted = 0

If your host provides an echo, the bypass cancel character should be set to the last character sent by the host when it echoes a line of text to the terminal.

If your host doesn't provide an echo, you probably don't need Bypass mode, so you should set the bypass cancel character to NU. See Section 2 for an explanation of the terminal's Bypass mode.

Syntax Example

```
Host: ^cNU:  
Setup: BYPASSCANCEL 10
```

Specifies the LF (Linefeed) character (ADE 10, encoded :) as the bypass cancel character.

Related Command

```
ENTER BYPASS MODE
```

SET COLOR COPIER REPAINT

Specifies the number of times the Tektronix 4692 Color Graphics Copier repaints an image. (Can be saved in nonvolatile memory.)

Host Syntax

```
^cQT repaint-count
```

Setup Syntax

```
HCREPAINT repaint-count
```

repaint-count: integer; specifies the number of times the image is transferred to the copier. Must be in the range 0 through 4 (0 defaults to 1).

Defaults: Factory = 1
Omitted = 1

This command specifies how many times the terminal transmits an image to the 4692 Copier in the course of making a single copy. Transmitting the image several times concentrates ink more heavily on the copy. This is useful in preparing transparencies because it results in more intense colors.

The time required to make a copy is multiplied by the number of image passes. A *repaint-count* of 1 yields the fastest copy of an image. Copying the same image with a *repaint-count* of 4 takes four times as long.

This command affects only the 4692 Copier; it has no effect on the 4691, 4695, or monochrome copiers.

Syntax Example

```
Host: ^cQT4  
Setup: HCREPAINT 4
```

Repaints an image four times.

Related Commands

```
COPY  
HARDCOPY  
SELECT COLOR HARDCOPY IMAGE DENSITY  
SELECT HARDCOPY INTERFACE  
SET IMAGE ORIENTATION
```

SET COPY SIZE

Selects the image size (standard or small) for copies. (Can be saved in nonvolatile memory.)

Host Syntax

```
^cQA size
```

Setup Syntax

```
HCSIZE size
```

size: integer; selects the image size for the copy. Must be one of the following:

- 0 Selects standard size (8½" × 11)
- 1 Selects small size

Defaults: Factory = 0
Omitted = 0

On a Tektronix 4695 Color Graphics copier, this command allows you to select a small copy-size for either a screen copy or a dialog copy. On a small screen copy, the image is one-half the default size. On a Tektronix 4691 or 4692 Copier, only dialog area copies are affected.

On a small dialog area copy, the image is slightly larger than one-half the default size.

NOTE

Screen copies made on Tektronix 4691 and 4692 Copiers are not affected by this command, but dialog copies are.

Specifying the small size produces a faster copy, but only in eight colors: black, white, red, green, blue, cyan, magenta, and yellow.

Refer to the **HARDCOPY** command for additional information about screen and dialog area copies.

If you are using Column mode 132, the small copy size allows you to copy 132 columns on the same line. If you choose the standard copy size with Column mode 132, the extra 52 columns are wrapped to the next line.

This command has no effect on a monochrome copier — you always get the default copy size

Syntax Example

```
Host: ^cQA1  
Setup: HCSIZE 1
```

Selects the smaller copy size.

Related Commands

```
COPY  
HARDCOPY  
SELECT HARDCOPY INTERFACE
```

SET DIALOG AREA BUFFER SIZE

Specifies the number of lines available for storing text in the dialog area buffer. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcLB number-of-lines
```

Setup Syntax

```
DABUFFER number-of-lines
```

number-of-lines: integer; specifies the number of lines in the dialog area buffer. Must be in the range 2 through 32767.

Defaults: Factory = 49
Omitted = Error

If you make the dialog area buffer smaller than the dialog area, the terminal automatically shrinks the dialog area to match the dialog area buffer. See *Displaying Dialog Between a Host and a User* in Section 4 for a detailed discussion of how the dialog buffer works with graphics applications.

Specifying a large dialog area buffer uses volatile memory that could be used for other features such as macro definitions or panel definitions. See the discussion *Managing Program Memory* in Section 4 for an explanation of how the dialog area buffer size affects the availability of volatile memory.

The ANSI command TEKSTBM (Set Top and Bottom Margins) can divide the dialog area into scrolling and fixed regions. If you have fixed regions set up and you specify a dialog area buffer larger than the screen size (30 lines), the scrolling region's top and bottom margins are set to Lines 1 and 30.

Similarly, if you have used the ANSI command RM (Reset Modes) to select Origin mode Absolute and you specify a dialog area buffer larger than the screen size, the Origin mode is set to Relative. See Section 3 for a complete explanation of how the dialog buffer works with screen editing applications.

Syntax Example

```
Host: EcLBA>  
Setup: DABUFFER 30
```

Selects a dialog area buffer of 30 lines (encoded *A* >).

Related Commands

```
SET DIALOG AREA LINES  
SET DIALOG AREA VISIBILITY
```


SET DIALOG AREA COLOR MAP

Specifies the colors assigned to color indices in the dialog area. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcTF color-mixtures
```

Setup Syntax

```
DACMAP color-mixtures
```

color-mixtures: integer array; assigns a color mixture to one or more color indices for the dialog area.

Defaults: Factory = See Table 5-4
 Omitted = Error

The integers in the *color-mixtures* array are in groups of four called *quadruples*. There is one quadruple for each color index you define, and the array count is the number of quadruples times four (if you assign color mixtures to all eight dialog area indices, the array count will be 32). The first integer in each quadruple names a color index, while the following three integers define the color mixture for that color index.

The color mixture is specified in the HLS color coordinate system. The valid ranges for the first, second, and third coordinates are:

```
Hue      -32768 — 32767
Lightness 0 — 100
Saturation 0 — 100
```

The color assigned to Index 0 applies only to the character itself. For the dialog area background and character background, Index 0 always means "transparent."

The discussion *Displaying Colors* in Section 4 explains the concept of transparency and how the terminal displays colors. Also refer to Appendix E, the *Tektronix Color Standard*.

NOTE

The user can change the color mixtures in the dialog area color map from the keyboard by using the Interactive Color Interface. So, if your program needs to have specific colors assigned to particular indices, you will need to reissue the SET DIALOG COLOR MAP command to ensure that you get the colors you want.

The user can alter colors from the terminal keyboard by using the *Interactive Color Interface*. See the discussion *Displaying Colors* in Section 4 for more details.

Table 5-4 shows the factory default colors assigned to color indices when the terminal is shipped from the factory. The SET DIALOG AREA COLOR MAP command lets you change these assignments.

Syntax Example

```
Host: EcTF830F4020C2F4
Setup: DACMAP 3,0,100,0,2,0,50,100
```

Defines the color mixtures for two indices in the dialog area, white (HLS coordinates 0,100,0) for Index 3, and blue (HLS coordinates 0,50,100) for Index 2 (50 is encoded C2, and 100 is encoded F4). In the host example, the 8 that follows the opcode TF is the array count.

Related Command

SET SURFACE COLOR MAP

Table 5-4

DEFAULT DIALOG AREA COLOR INDICES

Color Index	Color Mixture	Color Coordinates		
		Hue	Lightness	Saturation
0	Black	0	0	0
1	White	0	100	0
2	Red	120	50	100
3	Green	240	50	100
4	Blue	0	50	100
5	Cyan	300	50	100
6	Magenta	60	50	100
7	Yellow	180	50	100

SET DIALOG AREA HARDCOPY ATTRIBUTES

Sets attributes for a copy initiated by the HARDCOPY command or the D Copy key. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcQL number-of-pages  
page-origin  
Ff-interpretation
```

Setup Syntax

```
HCDATTRIBUTES number-of-pages  
page-origin  
Ff-interpretation
```

number-of-pages: integer; specifies how many pages to copy. Must be in the range 0 to 32767 (0 means no change from the last setting).

Defaults: Factory = 1
Omitted = 1

page-origin: integer; specifies the copy's starting point. Must be one of the following:

- 0 First line on the screen
- 1 Top of the dialog area buffer
- 2 Bottom of the dialog area buffer

Defaults: Factory = 0
Omitted = 0

Ff-interpretation: integer; specifies how the terminal treats Form Feed (Ff) characters in the dialog area buffer. Must be one of the following:

- 0 Starts a new page every 60 lines, ignoring Ff
- 1 Starts a new page every 66 lines or when Ff appears in the text
- 2 Starts a new page only when Ff appears in the text

Defaults: Factory = 0
Omitted = 0

This command specifies how many pages of dialog to copy, the starting point, and how to treat Form Feeds. As a default, the terminal sends 60 lines of text to the copier, starting with the first visible line on the screen.

If *page-origin* is set to 0, the copy starts with the first line of text visible in the dialog area. If *page-origin* is set to 1, the copy starts with the first line of text in the dialog area buffer. If *page-origin* is set to 2, the copy starts as many pages up from the bottom as *number-of-pages* specifies, and copies from there back to the bottom of the dialog area buffer.

The *Ff-interpretation* parameter determines the page size. When *Ff-interpretation* is set to 0, each page is 60 lines of text, with three blank lines at the top and bottom of the page (the terminal ignores Ff). When *Ff-interpretation* is set to 1, the terminal starts a new page each time it encounters a Ff in the text, as well as after every 66 lines. When *Ff-interpretation* is set to 2, the terminal starts a new page only when it encounters a Ff in the text.

If you set *number-of-pages* to 1, *page-origin* to 1, *Ff-interpretation* to 2, and there are no Form Feeds in the text, then the copier prints the entire dialog area buffer with no page breaks (on the Tektronix 4695 Copier, this would be a continuous sheet of paper).

If, however, you set *number-of-pages* to 4, *Ff-interpretation* to 2, and have four Form Feeds at the beginning of the text on the screen, then the copier turns out only four blank sheets of paper.

For lines that are wider than 80 characters, the terminal sends the long lines without inserting a Cr; this allows you to use printers that can print wide columns.

For standard sized copies with line length greater than 80, the Tektronix 4691, 4692, and 4695 Color Copiers, which don't have wide column capability, automatically generate a CrLf and start printing on the next line. For small copies, all three copiers print 132 characters on the same line. (Copy size is selected with the SET COPY SIZE command).

Syntax Example

```
Host: EcQL211  
Setup: HCDATTRIBUTES 2,1,1
```

Specifies that, when you issue the HARDCOPY command, the terminal should copy two pages of the dialog area, beginning at the top of the dialog area buffer, starting a new page after each 60 lines of text or when Ff is encountered.

Related Command

HARDCOPY

SET DIALOG AREA INDEX

Specifies the color index for alphanumerical characters, character-cell background, and dialog area background. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcLI character-index
character-background-index
dialog-background-index
```

Setup Syntax

```
DAINDEX character-index
character-background-index
dialog-background-index
```

character-index: integer; specifies the color index of the characters displayed in the dialog area. Must be in the range 0 through 65535.

Defaults: Factory = 1
 Omitted = 0

character-background-index: integer; specifies the color index used for each character cell background. Must be in the range 0 through 65535. Index 0 specifies transparent.

Defaults: Factory = 0
 Omitted = 0

dialog-background-index: integer; specifies the color index of the dialog area background. Must be in the range 0 through 65535. Index 0 specifies transparent.

Defaults: Factory = 0
 Omitted = 0

The dialog area uses different color indices than those used in the graphics area. There are eight color indices available, corresponding to values 0 through 7; a value greater than 7 sets the color index to 7. The dialog area's background index specifies the dialog area's color before characters are written on it and after it is erased.

When the terminal displays a character in the dialog area, it also displays the character cell that encloses the character. The cell is displayed in the character background color, set with the *character-background-index* parameter.

When Index 0 is used for alphanumerical characters, it represents an opaque color just like Indices 1 through 7. However, when Index 0 is used for the character-cell background or the dialog area background, that background becomes transparent and whatever is behind it shows through. If both the character and dialog backgrounds are transparent, the dialog appears to be written on a piece of glass in front of the graphics area.

The ANSI SGR (SELECT GRAPHICS RENDITION) command gives you the same control over the dialog area and gives you additional control over text display — you can select underscoring, blinking, and reversed-video text. The ANSI mode TEKSCNM command can reverse colors in both the graphics area and dialog area.

Syntax Example

```
Host: EcLI321
Setup: DAINDEX 3,2,1
```

Sets the *character-index* to 3 (default value *green*), the *character-background-index* to 2 (default value *red*), and the *dialog-background-index* to 1 (default value *white*).

Related Commands

```
SET DIALOG AREA VISIBILITY
SET DIALOG AREA WRITING MODE
SGR (SELECT GRAPHICS RENDITION)1
TEKSCNM (SCREEN MODE)1
```

¹ These are ANSI commands, described in Section 3.

SET DIALOG AREA LINES**SET DIALOG AREA LINES**

Specifies the number of lines visible in the dialog area. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcLL number-of-lines
```

Setup Syntax

```
DALINES number-of-lines
```

number-of-lines: integer; specifies how many lines are in the dialog area. Must be in the range 2 through 30.

Defaults Factory = 30
Omitted = Error

If you make the dialog area larger than the dialog buffer (assuming both are less than 30 lines), the terminal expands the dialog buffer to be as large as the dialog area.

Syntax Example

```
Host: EcLL?
Setup: DALINES 15
```

Sets the dialog area to 15 lines (encoded ?).

Related Commands

```
SET DIALOG AREA BUFFER SIZE
SET DIALOG AREA VISIBILITY
```

SET DIALOG AREA VISIBILITY

Specifies whether the dialog area is visible. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcLV visibility-mode
```

Setup Syntax

```
DAVISIBILITY visibility-mode
```

visibility-mode: integer (keyword in Setup); sets the dialog area to be either invisible or visible. Must be one of the following:

<u>Host</u>	<u>Setup</u>	
0	no	Dialog area invisible
1	yes	Dialog area visible

Defaults: Factory = 1
Omitted = 1

This command serves the same purpose as the Dialog key — it determines whether the dialog area is visible or not.

If the dialog area is enabled but not visible, the terminal stores alphatext in the dialog area buffer even though the dialog area is not visible. When the dialog area is made visible, the alphatext in the dialog area buffer becomes visible, too. The terminal will automatically scroll the dialog area, if necessary, to put the cursor in view.

Related Commands

```
CLEAR DIALOG SCROLL
ENABLE DIALOG AREA
SET DIALOG AREA BUFFER SIZE
SET DIALOG AREA INDEX
SET DIALOG AREA LINES
SET DIALOG AREA WRITING MODE
SET 4014 ALPHATEXT SIZE
```

SET DIALOG AREA WRITING MODE

Controls how the terminal displays Underscore and Space characters sent to the terminal screen. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcLM writing-mode
```

Setup Syntax

```
DAMODE writing-mode
```

writing-mode: integer (keyword in Setup); specifies the way the terminal treats the Space (␣) and Underscore (⏟) characters. Must be one of the following:

Host	Setup	
0	replace	Replaces characters
1	overstrike	Overwrites characters

Defaults: Factory = 0
Omitted = 0

Use this command with screen editing programs that rely on a printer's overstrike capability to create underscoring. TEKORM allows you to display underscoring in formatted files so that it looks the same on the screen as it does on a hard copy. Screen editing programs that turn underscoring on and off with the ANSI command SGR (SELECT GRAPHIC RENDITION) do not need to use the TEKORM command to emulate underscoring on the terminal screen.

When Overstrike/Replace mode is set to *replace* (which is the terminal's factory default), the Space and Underscore characters overwrite other characters², as they normally do. When Overstrike/Replace mode is set to *overstrike*, the terminal treats Space and Underscore in the same way as a printer does — the Underscore character underlines the current character and the Space character just moves the cursor forward without erasing characters. (On the screen, however, the Space character erases underscores).

Alphatext displayed in the graphics area is not affected by this command. SET GRAPHICS AREA WRITING MODE sets overstrike capability for graphics area alphatext.

The ANSI-style command TEKORM (OVERSTRIKE/REPLACE MODE) also controls the Space and Underscore characters in the same way as the SET DIALOG AREA WRITING MODE command.

Related Commands

SET DIALOG AREA INDEX
SET GRAPHICS AREA WRITING MODE
SGR (SELECT GRAPHIC RENDITION)¹
TEKORM (OVERSTRIKE/REPLACE MODE)¹

¹ These are ANSI commands, described in Section 3.

² Unless Insert/Replace mode (IRM) is set to *insert*.

SET ECHO**SET ECHO**

Specifies whether the terminal echoes characters it transmits to the host. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcKE echo-mode
```

Setup Syntax

```
ECHO echo-mode
```

echo-mode: integer (keyword in Setup); specifies whether the terminal provides a local echo. Must be one of the following:

Host	Setup	
0	no	No local echo — the terminal does not echo
1	yes	Local echo — the terminal echoes

Defaults: Factory = 0

Omitted = 1

When a character is typed on the keyboard, the character displayed on the screen is an echoed character. Some hosts provide an echo, and some do not. By default, the terminal does not provide an echo.

If the host provides the echo and the terminal is set to echo too, you will need to disable local echo; otherwise, characters typed on the keyboard will be displayed twice, LLIKKKEE TTHHIISS. If the host does not provide the echo, set the terminal for *local echo*; otherwise, characters typed on the keyboard will not appear on the screen.

In Setup (and in Local mode) the terminal always provides the echo.

The ANSI command SRM (SEND/RECEIVE MODE) can also be used to enable and disable local echo.

Related Commands

LOCAL

SRM (SEND/RECEIVE MODE)¹

¹ This is an ANSI command, described in Section 3.

SET EDIT CHARACTERS

Specifies the special text-editing characters used in the dialog area while in Setup. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcKZ character-delete
line-delete
literal
```

Setup Syntax

```
EDITCHARS character-delete
line-delete
literal
```

character-delete: integer (small integer in Setup); specifies the key that erases the character to the left of the cursor.

Defaults: Factory = 127 (Pr — the Rub Out key)

Omitted = Unchanged

line-delete: integer (small integer in Setup); specifies the key used in Setup to delete the current line.

Defaults: Factory = 24 (cN — the Ctrl-X key combination)

Omitted = Unchanged

literal: integer (small integer in Setup); specifies the character used just before an editing character to suspend its control action and print it as text.

Defaults: Factory = 126 (~)

Omitted = Unchanged

This command assigns two keys to act as editing keys in the dialog area, and one key to act as the literal character.

The literal character allows you to enter editing characters as characters for a Setup command, rather than for their editing function. Just precede the editing character with the literal character. Use the literal character to enter cR as well. The literal character affects only the character that immediately follows.

Syntax Example

Host: EcKZG?A8G>

Setup: EDITCHARS **Rub Out**,24,~

Assigns the Rub Out key (ADE 127, encoded as G?) as the delete-character key, the Ctrl-X key combination (ADE 24, encoded as A8) as the line-delete characters, and the ≈ (ADE 126, encoded as G>) as the literal character.

Related Commands

DEFINE MACRO

DEFINE NONVOLATILE MACRO

SET EOF STRING

Specifies the string the host sends to the terminal to indicate the end of a file. (Can be saved in nonvolatile memory.)

Host Syntax

```
^cNE EOF-string
```

Setup Syntax

```
EOFSTRING EOF-string
```

EOF-string: integer array (delimited string in Setup); specifies the ASCII characters in the EOF string. Each integer in the array must be in the range from 0 through 127.

Defaults: Factory = Empty array
 Omitted = Empty array

This command defines the terminal's host port end-of-file string. When the terminal receives this string from the host during a COPY operation, it knows that the end of a file transfer has been reached and terminates the COPY operation.

The EOF string may not contain more than ten characters, and should be set to match whatever string your host actually sends at the end of a file.

Syntax Example

```
Host: ^cNE3E8E9E:
Setup: EOFSTRING /XYZ/
```

Specifies XYZ as the EOF string. In the host example, The 3 following the escape sequence NE is the array count; X, Y, and Z are encoded E8, E9, and E:, respectively.

Related Commands

COPY
 PLOT

SET EOL STRING

Specifies the terminal's end-of-line string. (Can be saved in nonvolatile memory.)

Host Syntax

```
^cNT EOL-string
```

Setup Syntax

```
EOLSTRING EOL-string
```

EOL-string: integer array (delimited string in Setup); specifies the ASCII characters in the EOL string. Each integer in the array must be in the range from 0 through 127.

Defaults: Factory = 13 (CR)
 Omitted = Empty array

The end-of-line string usually consists of the single character CR (ADE 13), but it can contain up to two ASCII characters.

The terminal sends the EOL string at the end of all 4100-style reports it sends to the host (the EOL-string is not sent as part of ANSI reports, the answerback string, or the 4010 Status Report). The terminal also sends an EOL string to break up any report longer than 72 characters and to indicate the individual parts of the Error Report. See the discussion of the EOL string under *Reports* at the end of this section.

Syntax Example

```
Host: ^cNT1=
Setup: EOLSTRING /~CR/
```

Causes the terminal to treat CR (ADE 13, encoded =) as an end-of-line string. In the Setup example, The ~ (tilde) preceding the CR is the literal edit character (see the SET EDIT CHARACTERS command for details of its use).

Related Commands

REPORT ERRORS
 REPORT TERMINAL SETTINGS

SET EOM CHARACTERS

Specifies either of two characters that the terminal can use to mark the end of a line of text in data sent to the host. (Can be saved in nonvolatile memory.)

Host Syntax

```
^cNC first-EOM-character  
second-EOM-character
```

Setup Syntax

```
EOMCHARS first-EOM-character  
second-EOM-character
```

first-EOM-character: integer (small integer in Setup); specifies the ADE of one EOM character. Must be in the range from 0 through 127.

Defaults: Factory = 13 (C_R)
Omitted = 0 (N_U)

second-EOM-character: integer (small integer in Setup); specifies the ADE of another EOM character. Must be in the range from 0 through 127.

Defaults: Factory = 10 (L_F)
Omitted = 0 (N_U)

Typically, you set the EOM characters to whatever characters terminate a command line to your host. If you set both characters to N_U, the terminal will not use the transmit delay for characters typed from the keyboard.

If the terminal is in Prompt mode when the operator types either EOM character, the terminal stops transmitting until it receives a prompt string from the host *and* the transmit delay expires (see the SET TRANSMIT DELAY command). Then it sends the next line of text.

If the terminal is not in Prompt mode when the operator types either EOM character, the terminal stops transmitting only until the transmit delay expires before sending further information.

Syntax Example

```
Host: ^cNC = :  
Setup: EOMCHARS 13,10
```

Defines C_R (ADE 13, encoded =) and L_F (ADE 10, encoded .) as EOM characters.

Related Commands

```
PROMPT MODE  
SET TRANSMIT DELAY
```

SET ERROR THRESHOLD

Specifies the levels of error messages the terminal displays.

Host Syntax

```
^cKT error-threshold-level
```

Setup Syntax

```
ERRORLEVEL error-threshold-level
```

error-threshold-level: integer; specifies the lowest error level displayed. Must be one of the following:

- 0 Displays all messages, warnings, errors, and terminal failure messages
- 1 Displays warnings, errors, and terminal failure messages
- 2 Displays errors and terminal failure messages
- 3 Displays terminal failure messages
- 4 No messages, warnings, errors, or terminal failure messages displayed

Defaults: Factory = 2
Omitted = 0

This command determines the level of error messages that are displayed on the screen; it has no effect on which errors are reported to the host. When you issue a REPORT ERRORS command, the terminal records the eight most recent error messages and transmits them in response to a REPORT ERRORS command, regardless of what threshold level you set.

See Appendix B for descriptions of all error messages.

Related Command

```
REPORT ERRORS
```


SET FLAGGING MODE

Specifies the kind of flagging the terminal uses. (Can be saved in nonvolatile memory.)

Host Syntax

```
^cNF flagging-mode
```

Setup Syntax

```
FLAGGING flagging-mode
```

flagging-mode: integer (keyword in Setup); selects a flagging mode. Must be one of the following:

Host	Setup	
0	none	No flagging
1	input	DC1/DC3 flagging on input from the host
2	output	DC1/DC3 flagging on output to the host
3	in/out	DC1/DC3 flagging on both input from and output to the host
4	DTR/CTS	DTR/CTS flagging

Defaults: Factory = 0
 Omitted = 0

Flagging controls the flow of data between the terminal and host by providing a means for either device to signal the other whenever it's ready to send or receive data. This prevents the input queues of each from overflowing, thus losing data. This setting must match the flagging scheme used by your host. See Section 2 for more details.

If the host uses the DC1/DC3 scheme, users can use the Ctrl-S and Ctrl-Q key combinations to stop and start output from the host.

SET GIN CURSOR COLOR

Specifies the color mixture for the GIN cursor. (Can be saved in nonvolatile memory.)

Host Syntax

```
^cTC first-color-coordinate  

    second-color-coordinate  

    third-color-coordinate
```

Setup Syntax

```
GCURSOR first-color-coordinate  

    second-color-coordinate  

    third-color-coordinate
```

first-color-coordinate: integer; selects a value for hue. Must be in the range 0 through 360°.

Defaults: Factory = 0
 Omitted = 0

second-color-coordinate: integer; selects a value for lightness. Must be in the range 0 through 100%.

Defaults: Factory = 100
 Omitted = 0

third-color-coordinate: integer; selects a value for saturation. Must be in the range 0 through 100%.

Defaults: Factory = 0
 Omitted = 0

This command specifies the color mixture for the GIN cursor using the HLS coordinate system.

Syntax Example

```
Host: ^cTCK4C2F4  

    Setup: GCURSOR 180,50,100
```

Specifies that the GIN cursor should be yellow (HLS color coordinates 180,50,100, encoded *K4 C2 F4*).

SET GIN CURSOR SPEED

Determines how fast the GIN cursor moves across the screen when the Joydisk is pressed. (Can be saved in nonvolatile memory.)

Host Syntax

```
^cIJ normal-speed  
      shifted-speed
```

Setup Syntax

```
GSPEED normal-speed  
        shifted-speed
```

normal-speed: integer; determines the speed of the GIN cursor when the Joydisk is pressed. Must be in the range 1 through 10.

Defaults: Factory = 10
 Omitted = 1

shifted-speed: integer; determines the speed of the GIN cursor when both the Joydisk and the Shift key are pressed. Must be in the range 1 through 10.

Defaults: Factory = 1
 Omitted = 1

You can choose the cursor speed from a scale of 1 to 10, with 1 being the slowest speed and 10 being the highest. If you specify a value less than 1, the terminal interprets it as 1. If you specify a value greater than 10, the terminal interprets it as 10.

SET GRAPHICS AREA WRITING MODE

Specifies whether the terminal overwrites or replaces a character or marker in the graphics area. (Can be saved in nonvolatile memory.)

Host Syntax

```
^cMG writing-mode
```

Setup Syntax

```
GAMODE writing-mode
```

writing-mode: integer (keyword in Setup); must be one of the following:

<u>Host</u>	<u>Setup</u>	
0	replace	Specifies replace
1	overstrike	Specifies overstrike

Defaults: Factory = 1
 Omitted = 0

This command affects the way the terminal displays alphanext, markers, and graphtext in the graphics area.

If *writing-mode* is set to *overstrike*, characters can be superimposed on other characters. That is, the terminal can write a new character into the same character cell occupied by an existing character without first erasing the existing character. You can use overstrike to underline characters with _ (the Underscore character).

If *writing-mode* is set to *replace*, the terminal does not superimpose characters. Before the terminal writes a new character at the location of an existing character, the existing character is erased (with the background index specified in the SET VIEW ATTRIBUTES command). The terminal then displays the new character.

The SET GRAPHICS AREA WRITING MODE command affects alphanext in the graphics area, graphtext, and markers.

Related Commands

```
DRAW MARKER  
GRAPHIC TEXT  
SET BACKGROUND INDICES  
SET DIALOG AREA WRITING MODE  
SET VIEW ATTRIBUTES
```

SET GRAPHTEXT CHARACTER PATH

Selects a direction (right, left, up, down) to move after writing a graphtext character.

Host Syntax

```
EcMN direction
```

Setup Syntax

```
GTPATH direction
```

direction: integer (keyword in Setup); specifies which direction graphtext characters are written. Must be one of the following:

Host	Setup	
0	right	Equal to rotation angle
1	left	180° greater than rotation angle
2	up	90° greater than rotation angle
3	down	90° less than rotation angle

Defaults: Factory = 0
 Omitted = 0

The effect of the character path setting is relative to the rotation angle specified in SET GRAPHTEXT ROTATION. For example, if the rotation angle is 90° and the character path is set to *right*, the characters are written towards the top of the screen. The sense of the keywords (right, up, left, and down) applies to the character path at 0° rotation. Figure 5-7 shows how the terminal displays the string "ABC" using the four different directions for character path. Note that graphtext rotation is 0° for all examples in Figure 5-7.

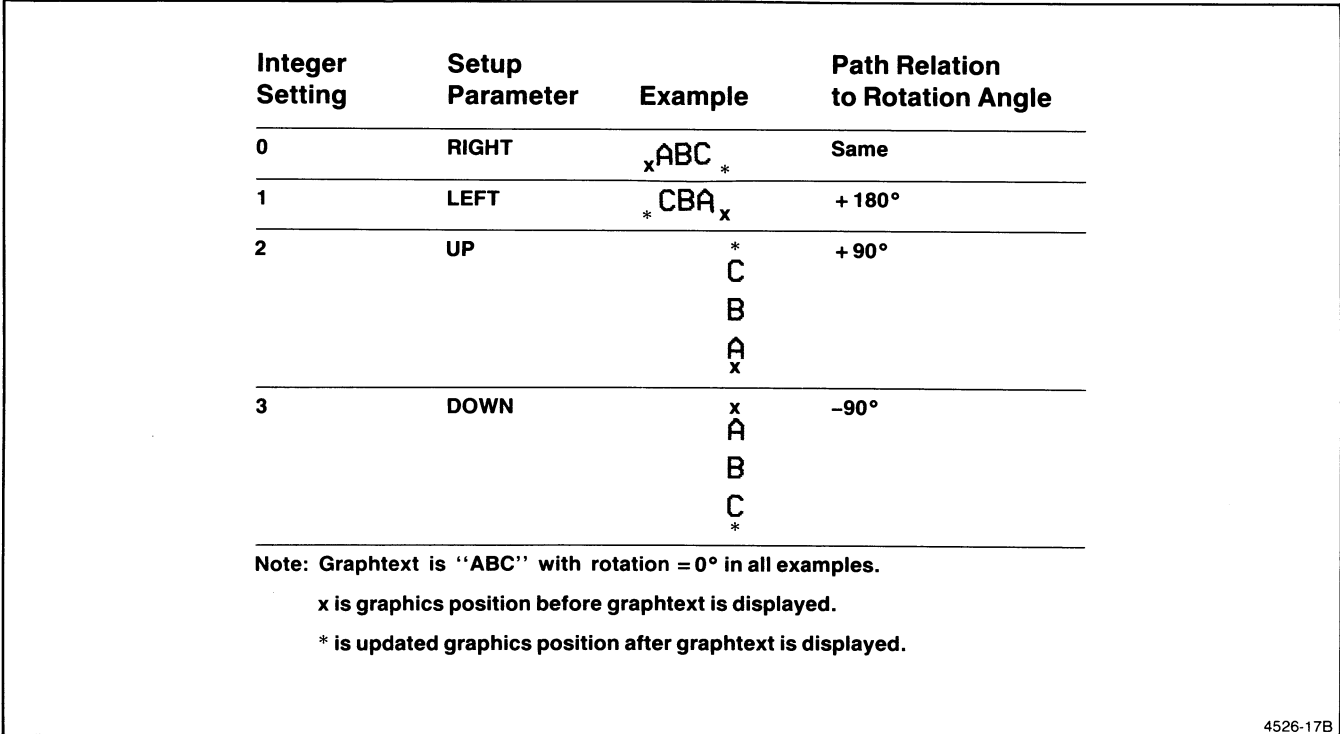
Syntax Example

Host: EcMN2
 Setup: GTPATH UP

Writes each graphtext character above the previous one at 0° rotation.

Related Commands

GRAPHIC TEXT
 SET GRAPHTEXT ROTATION



4526-17B

Figure 5-7. Character Path Settings.

SET GRAPHTEXT ROTATION

Specifies the rotation angle (in degrees) for graphtext.

Host Syntax

```

EcMR mantissa
    power-of-two
    
```

Setup Syntax

```

GTROTATION mantissa
    power-of-two
    
```

mantissa: integer; indicates the rotation angle. The valid values for for your terminal are 0, 90, 180, and 270. If you enter other values, the terminal will round them off to the nearest valid value as shown in Table 5-5.

Defaults: Factory = 0
 Omitted = 0

power-of-two: integer; gives the power of two by which the mantissa is multiplied. This is usually 0.

Defaults: Factory = (none)
 Omitted = 0

These two parameters are included for compatibility with other Tektronix terminals which can rotate graphtext to any angle.

Figure 5-8 shows the effect of rotating graphtext.

Syntax Example

```

Host: EcMRE:0
Setup: GTROTATION 90,0
    
```

Specifies a clockwise rotation of 90° for the graphtext string. In the host example, 90 is encoded *E*::; 0 is encoded *0*. See *Host Parameters* at the beginning of this section for a discussion of encoding real parameters.

Related Commands

- GRAPHIC TEXT
- SET GRAPHTEXT CHARACTER PATH

Table 5-5
GRAPHTEXT CHARACTER ROTATION

Specified Rotation	Actual Rotation
0.0 to 45.0°	0°
45.0 to 135.0°	90°
135.0 to 225.0°	180°
225.0 to 315.0°	270°
315.0 to 360.0°	0°

4526-18A

* is updated graphics position after graphtext is displayed.
 x is graphics position before graphtext is displayed.

Note: Graphtext is "ABC" and character path is in all examples.

270	*ABC*
180	*ABC*
90	*ABC
0	*ABC*

Setting Example

Figure 5-8. Graphtext Rotation Examples.

SET GRAPHTEXT SIZE

Sets the size of graphtext.

Host Syntax

```

EcMC width
      height
      spacing
  
```

Setup Syntax

```

GTSIZE width
        height
        spacing
  
```

width: integer; unused (can be omitted in Setup). Valid range is 0 through 4095. (This parameter is provided for compatibility with other Tektronix terminals.)

Defaults: Factory = 43 (TSU)
Omitted = Depends on *height* value

height: integer; specifies the height (in terminal space units) of a graphtext character. Must be in the range 0 through 4095; 0 specifies the default value.

Defaults: Factory = 61 (TSU)
Omitted = 61 (TSU)

spacing: integer; unused (can be omitted in Setup). Valid range is 0 through 4095. (This parameter is provided for compatibility with other Tektronix terminals.)

Defaults: Factory = 9 (TSU)
Omitted = Depends on *height* value

NOTE

You may want to include a reasonable value for the unused parameters if you want your application to run on other Tektronix terminals; if you specify too narrow a width or spacing, characters may be unreadable on other terminals.

The terminal displays graphtext in several fixed sizes, according to the height parameter you specify. The width and spacing parameters are accepted but ignored for graphtext. Each fixed size corresponds to a range of *height* values. The smallest available size displays an uppercase letter as five pixels wide and seven pixels high; the other available sizes are integer multiples of the smallest size. Table 5-6 gives the height ranges (in terminal space units) that yield the first three character sizes available.

Syntax Example

```

Host:  EcC?E9=
Setup: GTSIZE 63,89,13
  
```

Specifies a character size 89 terminal space units high (encoded E9), 63 terminal spaces wide (encoded C?), with 13 terminal space units (encoded =) between characters. This results in a displayed size of 10x14 pixels.

Note that the width and spacing parameters are not used by your 4105 terminal; coding them allows your application to display readable, well-proportioned graphtext on other Tektronix terminals.

Related Commands

```

GRAPHIC TEXT
SET GRAPHTEXT CHARACTER PATH
  
```

Table 5-6
GRAPHTEXT SIZES^a

Specified Height (TSU)	Resulting Size (Pixels)
1 — 88	5 × 7
89 — 146	10 × 14
147 — 205	15 × 21

^a These examples assume you've used the default window size.

SET HARDCOPY MONOCHROME ATTRIBUTES

Specifies the line termination (C_R or $C_R^L^F$) that the terminal sends to a monochrome printer. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcQE monochrome-attributes
```

Setup Syntax

```
HCMONOCHROME monochrome-attributes
```

monochrome-attributes: integer array (integer in Setup); specifies the line termination used in data sent to monochrome copiers. The array count in host syntax is always 1. Valid values are:

- 0 Sends just a C_R at the end of a line
- 1 Sends a $C_R^L^F$ combination at the end of a line

Defaults: Factory = 1
Omitted = 0

Use the SET HARDCOPY MONOCHROME ATTRIBUTES command to make the terminal send the appropriate line endings for your monochrome printer. This command affects copies made on either Centronics-style monochrome printers or monochrome graphics printers.

Your terminal usually sends a Carriage Return/Line Feed combination at the end of each line sent to a printer. Depending on switch settings on your printer, it may expect just a Carriage Return instead. If your terminal sends the wrong line endings for the printer, it can cause one of two problems:

- If the printer expects Line Feeds and the terminal doesn't send any, all your lines of text will print on the same line, resulting in a single unreadable black line.
- If the printer expects just Carriage Returns and the terminal sends extra Line Feeds, the copies you make will have an extra blank line following each line of characters (that is, single-spaced text will be double spaced, and graphics will have blank lines separating each line of graphics data).

Syntax Example

```
Host: EcQE10  
Setup: HCMONOCHROME 0
```

Instructs your terminal to send just a Carriage Return (C_R) at the end of each line. In the host example: the 1 following the opcode *QE* is the array count.

Related Commands

```
HARDCOPY  
SELECT HARDCOPY INTERFACE
```

SET IMAGE ORIENTATION

Specifies whether the long axis of an image aligns with the long or short axis of the paper. (Can be saved in nonvolatile memory.)

Host Syntax

E_cQO orientation

Setup Syntax

HCORIENT orientation

orientation: integer (keyword in Setup); specifies how an image is oriented on a copy. Must be one of the following:

Host	Setup	
0	horizontal	Long axis of image on long axis of paper
1	vbottom	Long axis of image on short axis of paper, with image at bottom of page
2	vcenter	Long axis of image on short axis of paper, with image centered vertically on page
3	vtop	Long axis of image on short axis of paper, with image at top of page

Defaults: Factory = 0
 Omitted = 0

On a 4691 or 4692 Copier, this command selects a horizontal or vertical orientation for the copied image. This command has no effect on copies made on the 4695 Color Graphics Copier.

At any of the vertical orientations (vbottom, vcenter, or vtop) the image size is reduced to fit on the narrow axis of the paper.

Syntax Example

Host: $\text{E}_c\text{QO}2$
 Setup: **HCORIENT vcenter**

Centers the image vertically, with its long axis aligned with the short axis of the paper (*orientation 2*).

Related Commands

COPY
 HARDCOPY
 SELECT COLOR HARDCOPY IMAGE DENSITY
 SELECT HARDCOPY INTERFACE
 SET COLOR COPIER REPAINT
 4010 HARDCOPY

SET KEY EXECUTE CHARACTER

Specifies the character used in key macros to specify whether subsequent characters should be processed by the host or by the terminal. (Can be saved in nonvolatile memory.)

Host Syntax

E_cKY key-execute-character

Setup Syntax

KEYEXCHAR key-execute-character

key-execute-character: integer (small integer in Setup); specifies the character. Valid values are in the range 0 through 127.

Defaults: Factory = 16 (PL)
 Omitted = 0 (NU)

This command selects the key-execute character, which is used during macro expansion and is part of a macro's definition.

The key-execute character acts as a toggle in the macro definition. That is, if the terminal is sending macros to the host, the key-execute character means "use what follows locally." If the terminal is currently using macros locally, the key-execute character means "send what follows to the host."

The key-execute character has this special effect only when the macro containing it is invoked by pressing a key. If the macro is invoked with an EXPAND MACRO command, the key-execute character is treated like any other character in the macro definition.

Syntax Example

Host: $\text{E}_c\text{KYA}8$
 Setup: **KEYEXCHAR 24**

Specifies the C_N character (ADE 24, encoded A8) as the key execute character.

Related Commands

DEFINE MACRO
 DEFINE NONVOLATILE MACRO
 EXPAND MACRO
 LEARN
 LEARN NONVOLATILE

SET LINE INDEX**SET LINE INDEX**

Specifies the color index for all subsequent lines, panel boundaries, and markers.

Host Syntax

```
EscML line-index
```

Setup Syntax

```
LINEINDEX line-index
```

line-index: integer; specifies the color index for lines, panel boundaries, and markers. Must be in the range 0 to 32767. (Values greater than 7 are set to 7.)

Defaults: Factory = 1
Omitted = 0

After this command is issued, all new lines, panel boundaries, and markers are drawn in the color specified by the index — until the terminal receives another SET LINE INDEX command that changes the index.

A host program assigns a color to an index with SET SURFACE COLOR MAP. A terminal user can use the Setup command CMAP or use the Interactive Color Interface to assign colors to indices.

Syntax Example

```
Host: EscML3
Setup: LINEINDEX 3
```

Specifies Index 3 (default value *green*) for subsequent lines, panel boundaries, and markers.

Related Commands

```
DRAW
DRAW MARKER
SET SURFACE COLOR MAP
```

SET LINE STYLE

Specifies the line style for subsequent lines and panel boundaries.

Host Syntax

```
EscMV line-style
```

Setup Syntax

```
LINESTYLE line-style
```

line-style: integer; selects a predefined line style. Must be in the range 0 through 7.

Defaults: Factory = 0 (solid line)
Omitted = 0 (solid line)

Figure 5-9 shows the line styles. The terminal displays all new lines and panel boundaries with the line style selected with this command or with the SET 4014 LINE STYLE command, whichever was last executed. Changing the line style has no effect on lines already drawn.

NOTE

While the dialog area is disabled, issuing a PAGE command resets the line style to 0.

Syntax Example

```
Host: EscMV1
Setup: LINESTYLE 1
```

Specifies a dotted line (*line style 1*) for subsequent lines and panel boundaries.

Related Commands

```
BEGIN PANEL BOUNDARY
DRAW
PAGE
SET 4014 LINE STYLE
```


Parameter	Line Style
0	—————
1
2	- - - - -
3	- - - - -
4	- - - - -
5
6	- - - - -
7	- - - - -

4526-19A

Figure 5-9. Line Styles.

SET MARKER TYPE

Selects the kind of marker to be drawn.

Host Syntax

E_cMM marker-number

Setup Syntax

MARKERTYPE marker-number

marker-number: integer; selects a predefined marker type. Must be in the range 0 through 10.

Defaults: Factory = 0
 Omitted = 0

When you change marker types, markers already displayed remain the same.

Figure 5-10 shows the predefined marker types.

Syntax Example

Host: **E_cMM:**
 Setup: **MARKERTYPE 10**

Specifies Marker Type 10 (encoded :).

Related Commands

ENTER MARKER MODE
 DRAW MARKER

Parameter	Marker Type	Parameter	Marker Type
0	·	6	□
1	+	7	◇
2	+	8	□
3	*	9	◇
4	○	10	⊠
5	×		

4526-42A

Figure 5-10. Marker Types.

SET PARITY**SET PARITY**

Specifies the kind of parity the terminal uses when it sends data to the host. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcNP parity-mode
```

Setup Syntax

```
PARITY parity-mode
```

parity-mode: integer (keyword in Setup); selects the kind of parity the terminal uses. Must be one of the following:

Host	Setup	
0	none	Parity bit set to 0
1	odd	Odd parity
2	even	Even parity
3	high	Parity bit set to 1
4	data	No parity; parity bit available for data

Defaults: Factory = 0
Omitted = 0

This command defines how the parity bit is set on characters sent to the host. This setting must match the parity checking scheme that the host uses on incoming data.

NOTE

The keyword none in the parity-mode parameter means the parity bit is set to 0. For no parity, use the keyword data.

When *parity-mode* is 1 (odd), the eighth bit in each character's binary ASCII representation is set so that the sum of all the bits in the character is odd. When *parity-mode* is 2 (even), the eighth bit is set so that the sum of all the bits in the character is even.

When *parity-mode* is 4 (no parity), the eighth bit can be data; the terminal, however, does not generate 8-bit data.

The terminal ignores the parity bit in characters it receives from the host.

SET PIXEL BEAM POSITION

Sets the position of the pixel beam in the pixel viewport.

Host Syntax

```
EcRH beam-position
```

Setup Syntax

```
PXPOSITION beam-position
```

beam-position: *xy*-coordinate; specifies the position of the pixel beam in the pixel viewport. Values for *x* range from 0 through 511; for *y*, from 0 through 359.

Defaults: Factory = 0,359
Omitted = 0,0

The pixel beam position is like an invisible cursor that marks where pixel writing will occur. Setting the pixel beam position in the pixel viewport establishes the starting point for a RASTER WRITE or RUNLENGTH WRITE command. Working from this point, these commands set pixel colors to create a visible image in the pixel viewport.

You set the pixel beam position relative to the lower-left corner of the pixel viewport. For instance, consider a pixel viewport that is 100 pixels square, with its lower-left corner located at 100,100 (in graphics memory) and its upper-right corner located at 200,200. To place the pixel beam in the center of the square, you must give beam position coordinates of 50,50 because the beam position is relative to the lower-left corner of the pixel viewport, not the graphics memory.

If you try to set the pixel beam to a position outside the pixel viewport, the terminal moves the beam to the nearest pixel inside the viewport. If, for instance, you give beam position coordinates of 150,150 and the pixel viewport is just 100 pixels square, the terminal positions the beam at 100,100, the upper-right corner of the viewport.

Syntax Example

```
Host: EcSppySPY  
Setup: PXPOSITION 100,100
```

Sets the pixel beam position at 100,100 (*xy*-coordinate encoded *SppySPY*).

Related Commands

```
RASTER WRITE  
RUNLENGTH WRITE  
SET PIXEL VIEWPORT
```

SET PIXEL VIEWPORT

Specifies the pixel viewport's size and position in graphics memory.

Host Syntax

```
EcRS lower-left  
upper-right
```

Setup Syntax

```
PXVIEWPORT lower-left  
upper-right
```

lower-left: xy-coordinate; specifies one corner of the pixel viewport. Values for x must be in the range 0 through 511; for y, 0 through 359.

Defaults: Factory = 0,0
Omitted = 0,0

upper-right: xy-coordinate; specifies the opposite corner of the pixel viewport. Values for x must be in the range 0 through 511; for y, 0 through 359.

Defaults: Factory = 479,359
Omitted = 0,0

The pixel viewport is a rectangular area in graphics memory. Commands that write pixels directly operate within the pixel viewport that was most recently defined by this command.

The coordinates specified in the *lower-left* and *upper-right* parameters may actually be the coordinates of any two diagonally opposite corners of the pixel viewport. The terminal will sort the two points to determine the correct upper and lower corners.

When you create a new pixel viewport, the terminal resets the pixel beam position to the upper-left corner of the pixel viewport.

You can set the viewport's x-axis to include the pixels 480 to 511, which are off the screen. A Level 0 warning will be generated, however.

Syntax Example

```
Host: EcRSSppySPY!pb!B  
Setup: PXVIEWPORT 100,100,200,200
```

Sets a pixel viewport with a lower-left corner of 100,100 (xy-coordinate encoded ^Spp_y^SP_Y) and an upper-right corner of 200,200 (xy-coordinate encoded !pb!B).

Related Commands

```
BEGIN PIXEL OPERATIONS  
SET PIXEL BEAM POSITION  
RASTER WRITE  
RUNLENGTH WRITE
```

SET PROMPT STRING

Specifies the string that initiates the terminal's Prompt mode. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcNS prompt-string
```

Setup Syntax

```
PROMPTSTRING prompt-string
```

prompt-string: integer array (delimited string in Setup syntax); specifies the characters in the string. Valid range for each character is ADE 0 — 127.

Defaults: Factory = Empty array
Omitted = Empty array

This command defines the string that the terminal recognizes as a prompt string from the host. When the terminal is in Prompt mode and receives the prompt string, the terminal waits a specified interval before sending the next line of data.

See Section 2 for an explanation of Prompt mode and other communications modes.

Syntax Example

```
Host: EcNS3F1F2F3  
Setup: PROMPTSTRING /abc/
```

Specifies the string *abc* as the terminal's prompt string. In the host example, the integer 3 after the opcode *NS* is an array count; the letter *a* is encoded *F1*, *b* is encoded *F2*, and *c* is encoded *F3*.

Related Commands

PROMPT MODE
SET TRANSMIT DELAY

SET QUEUE SIZE

Specifies the size (in bytes) of the terminal's input queue for host communications. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcNQ queue-size
```

Setup Syntax

```
QUEUESIZE queue-size
```

queue-size: integer; indicates the size in bytes of the input queue. Must be in the range 1 through 65535.

Defaults: Factory = 300
Omitted = Error

This command reserves part of program memory for use as an input queue for data coming from the host. Data from the host is stored in the input queue until it can be processed by the terminal. If the terminal is not using a flagging scheme, the input queue fills up, and the terminal simply discards incoming data until there is more room in the queue. See Section 2 for details.

Specifying a very small input queue may cause data to be lost when the input queue overflows. Specifying a large input queue uses program memory that could be used for other features such as macro definitions or panel definitions. See the discussion *How the Terminal's Memory Works* in Section 4 for an explanation of how the input queue size affects the availability of program memory.

Syntax Example

```
Host: EcNQx4  
Setup: QUEUESIZE 900
```

Sets an input queue size of 900 bytes (encoded *x4*).

Related Commands

SET BAUD RATES
SET FLAGGING MODE

SET SEGMENT POSITION

Moves the GIN cursor to a specified position in terminal space.

Host Syntax

```
EcSX segment-number
      position
```

Setup Syntax

```
SGPOSITION segment-number
            position
```

segment-number: integer; must be 0, which identifies the GIN cursor. This parameter is included for compatibility with other Tektronix terminals, which allow you to define and store segments and to use a segment as the GIN cursor. Must be 0, which identifies the crosshair cursor.

Defaults: Factory = (none)
 Omitted = 0

position: xy-coordinate; specifies the new location (in terminal space) of the GIN cursor. Both x and y must be in the range 0 through 4095.

Defaults: Factory = 0,0
 Omitted = 0,0

This command relocates the GIN cursor in terminal space and redraws the cursor at the new location.

Syntax Example

```
Host: EcSX1#\ }#]
Setup: SGPOSITION 1,500,500
```

Moves the GIN cursor to 500,500 (encoded #\ }#]).

Related Command

SET WINDOW

SET SNOOPY MODE

Specifies whether the terminal displays ASCII control characters.

Host Syntax

```
EcKS snoopy-mode
```

Setup Syntax

```
SNOOPY snoopy-mode
```

snoopy-mode: integer (keyword in Setup); must be one of the following:

Host	Setup	
0	no	Takes the terminal out of Snoopy mode
1	yes	Puts the terminal in Snoopy mode

Defaults: Factory = 0
 Omitted = 1

Control characters are the characters that normally control the terminal and aren't usually displayed on the screen. These are the characters shown in the first two columns of the code charts included in Appendix A.

When in Snoopy mode, the terminal displays control characters instead of executing them (except for D_1 and D_3 , which are executed but do not display, and L_F , which is displayed and causes a new display line). Characters that are normally filtered out of the host's data stream (such as a prompt sequence) are still filtered and not displayed.

While the terminal is in Snoopy mode, the terminal executes Setup commands only, and not commands sent from the host. As a result, only the user can take the terminal out of Snoopy mode — the host cannot do it. To terminate Snoopy mode, press the Cancel key, or enter Setup and issue *SNOOPY NO*.

Related Command

CANCEL

SET STOP BITS**SET STOP BITS**

Specifies the number of stop bits appended to each character the terminal transmits. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcNB number-of-stopbits
```

Setup Syntax

```
STOPBITS number-of-stopbits
```

number-of-stopbits: integer; specifies how many stop bits the terminal appends to each character it transmits. Valid values are 1 and 2.

Defaults: Factory = 1
Omitted = Error

While communicating with the host, the terminal transmits a character as a series of 10 or 11 bits. Here's the sequence of bits in the series:

1. The start bit (always a 0)
2. Seven data bits
3. The parity bit (depends on the SET PARITY command)
4. One or two stop bits (always sent as 1's — *number-of-stop-bits* specifies how many)

Refer to Section 2 for more information about how the terminal communicates with the host.

Related Command

SET PARITY

SET SURFACE COLOR MAP

Specifies the colors assigned to one or more indices in the graphics area. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcTG surface-number  
color-mixtures
```

Setup Syntax

```
CMAP surface-number  
color-mixtures
```

surface-number: integer; names the surface for which color mixtures are being defined. (Other Tektronix terminals have more than one surface, so the surface parameter is included here for compatibility.) Must have a value of 1, since there is just one surface defined on the 4105 terminal.

Defaults: Factory = 1
Omitted = Error TG11

color-mixtures: integer array; assigns color mixtures to one or more color indices. Table 5-7 lists the default color mixtures.

Defaults: Factory = See Table 5-7
Omitted = Error TG21

The integers in the *color-mixtures* array must be arranged in groups of four called *quadruples*. The first integer in each quadruple names a color index, while the following three integers specify the color mixture for that color index. (Remember that in host syntax, the first integer in the array is the array count. So, to define the color mixtures for two indices you would send first the array count of 8 followed by the two quadruples.)

The number of quadruples in the array depends on the number of indices you want to reset. You can define color mixtures for eight color indices (numbered 0 through 7). Thus you can include up to eight quadruples when issuing this command.

The color mixture is specified in the HLS color coordinate system. The valid ranges for the first, second, and third coordinates are:

```
Hue          -32768 — 32767
Lightness    0 — 100
Saturation   0 — 100
```

The effect of the SET SURFACE COLOR MAP command continues until superseded by another SET SURFACE COLOR MAP command or until modified from the keyboard through the Interactive Color Interface.

Use the SET DIALOG AREA COLOR MAP command to set the dialog area color map.

Syntax Example

Host: **EcTG14300C2F4**
 Setup: **CMAP 1,3,240,50,100**

Sets Index 3 to *green* (HLS color coordinates 240,50,100, encoded *O0C2F4*):. In the host example, the array count, 4, specifies the number of items in the color mixture array.

Note that the first parameter, 1, designates Surface 1, even though your terminal can display just one surface. This parameter is required by the SET SURFACE COLOR MAP command in order to be compatible with other Tektronix terminals that can display more than one surface.

Table 5-7

DEFAULT GRAPHICS AREA COLOR INDICES

Color Index	Color Mixture	Color Coordinates		
		Hue	Lightness	Saturation
0	Black	0	0	0
1	White	0	100	0
2	Red	120	50	100
3	Green	240	50	100
4	Blue	0	50	100
5	Cyan	300	50	100
6	Magenta	60	50	100
7	Yellow	180	50	100

SET TAB STOPS

Sets horizontal tab stops at the specified positions in the dialog buffer. (Can be saved in nonvolatile memory.)

Host Syntax

EcKB tab-positions

Setup Syntax

TABS tab-positions

tab-positions: integer array; specifies one or more tab stops. Valid values are integers from -2 to 132. Positive integers specify specific column positions; you can also specify 0 to clear all tabs, specify -1 (or the keyword *all* in Setup) to set tabs at every column, or specify -2 to reset tab stops to the factory default.

Defaults: Factory = Every eighth column (1, 9, 17, . . .)
 Omitted = 0

This command sets horizontal tab stops at the positions you indicate, and it clears tabs that you don't list.

There are two ANSI commands HTS (HORIZONTAL TAB SET) and TBC (TAB CLEAR) that perform similar functions. The ANSI commands CBT (CURSOR BACKWARD TAB) and CHT (CURSOR HORIZONTAL TAB) move the alphanumeric cursor to the default tab positions or to tab positions that you set.

Syntax Example

Host: **EcKB35:?**
 Setup: **TABS 5,10,15**

Sets tab stops at Columns 5, 10, and 15 (10 and 15 are encoded : and ?).

Related Commands

- CBT (CURSOR BACKWARD TAB)¹
- CHT (CURSOR HORIZONTAL TAB)¹
- HTS (HORIZONTAL TAB SET)¹
- TBC (TAB CLEAR)¹

¹ These are ANSI commands, described in Section 3.

SET TEXT INDEX

Specifies the color index for all text displayed in the graphics area.

Host Syntax

```
EcMT text-index
```

Setup Syntax

```
GTINDEX text-index
```

text-index: integer; specifies the color index for text in the graphics area. Valid values are in the range 0 through 65535.

Defaults: Factory = 1
Omitted = 0

This command sets the color index for all new text displayed in the graphics area. This includes all graphtext, and any alphanext displayed in the graphics area. This command does not change the color index of existing text.

Alphanext displayed in the dialog area is not affected by this command. Use the SET DIALOG AREA INDEX command for dialog area alphanext.

Syntax Example

```
Host: EcMT2  
Setup: GTINDEX 2
```

Sets the text color index to 2.

Related Commands

GRAPHIC TEXT
SET DIALOG AREA INDEX
SET VIEW ATTRIBUTES

SET TRANSMIT DELAY

Specifies the terminal's delay between transmitting lines of text. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcND transmit-delay
```

Setup Syntax

```
XMTDELAY transmit-delay
```

transmit-delay: integer; indicates the transmit delay in milliseconds; must be in the range 0 to 65535.

Defaults: Factory = 100
Omitted = 0

After the operator types an EOM character (or the terminal sends an EOL string as part of a report), the terminal pauses for a short time before resuming transmission. The length of this pause is set with the SET TRANSMIT DELAY command. This gives the host time to receive, verify, and process one line of text before receiving another. Because of the resolution of the terminal's internal timer, the actual delay time may be up to 33 milliseconds longer than the time specified by this command.

If the terminal is in Prompt mode, the terminal waits until it receives a prompt sequence *and* until the transmit delay has elapsed before it transmits another line of text.

Syntax Example

```
Host: EcNDL8  
Setup: XMTDELAY 200
```

Sets the transmit delay to 200 milliseconds (encoded *L8*).

Related Commands

ENTER BYPASS MODE
PROMPT MODE
SET EOM CHARACTERS

SET TRANSMIT RATE LIMIT

Specifies the effective transmit data rate limit. (Can be saved in nonvolatile memory.)

Host Syntax

```
EcNL rate-limit
```

Setup Syntax

```
XMTLIMIT rate-limit
```

rate-limit: integer; specifies the terminal's transmit rate limit; must be in the range 110 through 65535.

Defaults: Factory = 19200
Omitted = Error

In some circumstances the host may not be able to process information as fast as the terminal can send it. This command causes the terminal to pace its data transmission, spacing the characters apart so that it does not exceed the indicated rate limit.

Syntax Example

```
Host: EcNLR<  
Setup: XMTLIMIT 300
```

Sets the transmit baud rate limit to 300 bits per second (encoded *R* <).

Related Command

SET BAUD RATES

SET VIEW ATTRIBUTES

Selects the wipe index for the graphics area.

Host Syntax

```
EcRA surface-number  
wipe-index  
border-index
```

Setup Syntax

```
VIEWATTRIBUTES surface-number  
wipe-index  
border-index
```

surface-number: integer; identifies the surface. Must be 1 or -1. (This parameter provides compatibility with other Tektronix terminals that support more than one surface.)

Defaults: Factory = 1
Omitted = 0

wipe-index: integer; specifies the color index of the graphics area background. Must be in the range 0 through 65535.

Defaults: Factory = 0
Omitted = 0

border-index: unused. (This parameter provides compatibility with other Tektronix terminals.)

The wipe index is the color index to which the graphics area background is set when you erase it. Eight color indices are available. If you specify a wipe index greater than the highest numbered index available on your terminal, the terminal will default to its highest index. The color index you specify is one of the color indices defined by the SET SURFACE COLOR MAP command. When you change the background color, you do not see the new color until the screen is erased.

Syntax Example

```
Host: EcRA120  
Setup: VIEWATTRIBUTES 1,2,0
```

Changes the graphics area background color to Index 2 (default value *red*).

Related Command

SET SURFACE COLOR MAP

SET WINDOW**SET WINDOW**

Sets the boundaries of the current window in terminal space.

Host Syntax

```

ECRW first-corner
      second-corner

```

Setup Syntax

```

WINDOW first-corner
        second-corner

```

first-corner: xy-coordinate; specifies one corner of the window. Values for x and y must be in the range 0 through 4095.

Defaults: Factory = 0,0
Omitted = 0,0

second-corner: xy-coordinate; specifies the opposite corner of the window. Values for x and y must be in the range 0 through 4095.

Defaults: Factory = 4095,3132
Omitted = 0,0

A *window* is a rectangular region in terminal space whose contents are displayed on the screen.

The two xy-coordinates specify two opposite corners of the window. These can be any two opposite corners; the terminal sorts the two x-coordinates and the two y-coordinates in the proper order.

If you specify the same x-coordinate for both corners, the terminal will automatically assign an x-coordinate for the upper-right corner that gives the window the same aspect ratio (ratio of height-to-width) as the screen. Likewise, if you specify the same y-coordinate for both corners, the terminal will assign a y-coordinate that gives the window the same aspect ratio as the screen.

If you specify the same xy-coordinate for both corners, the terminal assigns the window its default dimensions.

Syntax Example

```

Host:  ECRWSPbySPL 5'|2Q
Setup:  WINDOW 50,100,2372,2800

```

Sets a window with the xy-coordinates 50,100 (encoded *S_Pby_SP_L*) as the first corner and 2372,2800 (encoded *5'|2Q*) as the second corner.

SET 4014 LINE STYLE

Specifies line styles compatible with Tektronix 4010 and 4110 Series terminals.

Host Syntax

```
Ec line-style-code
```

line-style-code: single character; specifies one of the line styles shown in Figure 5-11.

Defaults: Factory = \ (solid lines)
 Omitted = (none)

This command does the same thing as the SET LINE STYLE command. The line style for lines, markers, and panel boundaries is set by either SET LINE STYLE or SET 4014 LINE STYLE, whichever occurred most recently.

SET 4014 LINE STYLE sets line styles that are available on other Tektronix terminals. This lets you emulate other terminal's displays.

Codes *h* through *o* indicate line styles that are displayed with a defocused beam on Tektronix 4014, 4016, and 4114 Terminals. The 4105 terminal doesn't defocus these lines.

Related Command

SET LINE STYLE

Character	Line Style	Emulated Terminals
\	—————	4014/4016
a	4014/4016
b	- - - - -	4014/4016
c	- - - - -	4014/4016
d	- - - - -	4014/4016
e	4112/4113/4114
f	- - - - -	4112/4113/4114
g	- - - - -	4112/4113/4114
h	—————	4014/4016/4114
i	4014/4016/4114
j	- - - - -	4014/4016/4114
k	- - - - -	4014/4016/4114
l	- - - - -	4014/4016/4114
m	4014/4016/4114
n	- - - - -	4014/4016/4114
o	- - - - -	4014/4016/4114

4893-16

Figure 5-11. 4014 Line Styles.

STATUS**STATUS**

Displays the current settings for a command or cluster of commands.

Setup Syntax

```
STATUS name
```

name: string; the command name or command cluster name for which you want the current parameter values. To display the valid command names and command cluster names, enter **STATUS**.

Defaults: Factory = (none)
Omitted = All commands

Give a specific Setup name or host escape sequence to request status information about a specific command's settings (if there are any). If there is no status message for the command, try requesting the status of the cluster the command belongs to.

When you request the status of a cluster of commands, the terminal displays the settings of all the commands in the cluster. The cluster names are:

- ANSI
- Communications
- Dialog
- General
- Graphics
- Hardcopy
- Keyboard
- Pixels

You can also use three special names, which are part of the *general* cluster. These are:

- Memoryblocks
- Version
- Terminal

When you request the status of *memoryblocks*, the terminal displays (1) the total number of memory blocks available in volatile memory, and (2) the largest contiguous group of memory blocks available in volatile memory (one memory block contains 16 bytes). See the discussion *Managing Program Memory* in Section 4 for an explanation of how the terminal's features use memory.

When you request the status of *version*, the terminal displays its standard firmware version number.

When you request the status of *terminal*, the terminal displays its model number.

4010 HARDCOPY

Generates a hard copy of the entire screen.

Host Syntax

```
ESC^B
```

This command has the same effect as pressing the S Copy key.

To copy only the graphics area, press the Dialog key to make the dialog area invisible. After the copy starts, you can press the Dialog key to make the dialog area visible. This lets you work in the dialog area while the copy is being made.

Related Commands

HARDCOPY
SELECT HARDCOPY INTERFACE

REPORTS

The terminal sends reports to the host in response to commands sent from the host or in response to users' input during GIN operations. Each report is made up of report parameters, which use an encoding scheme similar to the scheme used for sending host parameters in commands.

This part of Section 5 describes the report parameters and the reports that the terminal sends to the host.

REPORT PARAMETERS

Report parameters are the types of data that the terminal sends in its reports to the host. Like command parameters, there are three basic report parameter types — the character, the integer, and the xy-coordinate — and two complex variations — the string and the integer report array.

Character Report Parameters

A *character report parameter* is an ASCII character whose ADE is in the range 0 through 127.

Integer Report Parameters

An *integer report parameter* is a sequence of three ASCII characters that represent an integer number in encoded form. The first two characters sent are Hi-I characters followed by the Lo-I character. The terminal always sends all three characters.

NOTE

The encoding scheme that the terminal uses for integer parameter reports is different than the encoding scheme that your program must use when it issues integer parameters in 4100-style commands. The example included here shows how the terminal encodes integer report parameters.

This discussion describes the encoded *integer report parameters* that the terminal uses when it sends 4100-style reports to the host. Note that, when you include integer parameters in 4100-style commands you send from the host, you must use *integer parameters*, which use a different encoding scheme. See the discussion *Integer Parameters* at the beginning of this section and the subroutine for encoding integer parameters in *Report Decoding Subroutines* in Section 6 to see how to encode integer parameters.

For example, to decode the integer report "M-", your decoding routine must first subtract 32 from each character's ADE. For example:

	ASCII Character	ADE	Difference (ADE - 32)
1st Hi-I	"	34	2
2nd Hi-I	M	77	45
Lo-I	-	45	13

Use the differences to calculate the decimal integer value as follows:

- Multiply the first Hi-I difference by 1024 and save the result:
 $2 * 1024 = 2048$
- Multiply the second Hi-I difference by 16 and save the result:
 $45 * 16 = 720$
- Find the modulo 16 value of the Lo-I difference and save the result:
 $13 \text{ mod } 16 = 13$
- Add the three results and save the sum:
 $2048 + 720 + 13 = 2781$
- Divide the Lo-I difference by 16 and save the integer part of the quotient:
 $13 \div 16 = 0$
Then find the quotient's modulo 2 value to determine the decoded integer's sign. If the value equals 0, the integer is negative:
 $0 \text{ mod } 2 = 0$
- Give the sign from Step 5 to the sum from Step 4; this yields the decimal integer:
 -2781

You will find a sample FORTRAN subroutine for decoding integer report parameters in Section 6.

REPORTS**XY-Coordinate Report Parameters**

An *xy-coordinate report parameter* (usually called just *xy-report*) represents in encoded form the 12-bit precision x- and y- coordinate values the terminal sends to the host. The terminal reports xy-coordinates to the host as five ASCII characters. All five are always sent in this sequence:

Hi-Y
Extra
Lo-Y
Hi-X
Lo-X

To decode the xy-report, your program must first subtract 32 from each character's ADE. For example:

	ASCII Character	ADE	Difference (ADE - 32)
<i>Hi-Y</i>	'	39	7
<i>Extra</i>	!	33	1
<i>Lo-Y</i>	:	58	26
<i>Hi-X</i>	sp	32	0
<i>Lo-X</i>	-	45	13

You then use the differences to calculate the decimal xy-coordinate values as follows:

1. Multiply the Hi-X difference by 128 and save the result:

$$0 * 128 = 0$$

2. Multiply the Lo-X difference by 4 and save the result:

$$13 * 4 = 52$$

3. Find the modulo 4 value of the Extra difference and save the result:

$$1 \bmod 4 = 1$$

4. Add the three results; the sum is the x-coordinate:

$$0 + 52 + 1 = 53$$

5. Multiply the Hi-Y difference by 128 and save the result:

$$7 * 128 = 896$$

6. Multiply the Lo-Y difference by 4 and save the result:

$$26 * 4 = 104$$

7. Divide the Extra difference by 4 and save the integer part of the quotient:

$$1 \div 4 = 0$$

Then find the modulo 4 value of the quotient, and save the result:

$$0 \bmod 4 = 0$$

8. Add the three results; the sum is the y-coordinate:

$$896 + 104 + 0 = 1000$$

Refer to Section 6 for a sample subroutine that decodes xy-reports.

4010 XY-Report Parameters

The terminal sends 4010 xy-reports (10-bit precision coordinates) to the host as four ASCII characters. All four are always sent using the following format:

Hi-X
Lo-X
Hi-Y
Lo-Y

Notice that this format has one less character than the xy-report format, and that the terminal sends the x and y characters in reverse order (x's first instead of y's).

To decode a 4010 xy-report, your program must first subtract 32 from each ADE, just as it did for the xy-report. For example:

	ASCII Character	ADE	Difference (ADE - 32)
<i>Hi-X</i>	'	39	7
<i>Lo-X</i>	:	58	26
<i>Hi-Y</i>	/	47	15
<i>Lo-Y</i>	4	52	20

Use the differences for each x and y to calculate the two decimal xy-coordinates as follows:

1. Multiply the Hi-X difference by decimal 128 and save the result:

$$7 * 128 = 896$$

2. Multiply the Lo-X difference by decimal 4 and save the result:

$$26 * 4 = 104$$

3. Add the results; the sum is the x-coordinate:

$$896 + 104 = 1000$$

4. Multiply the Hi-Y difference by 128 and save the result:

$$15 * 128 = 1920$$

5. Multiply the Lo-Y difference by 4 and save the result:

$$20 * 4 = 80$$

6. Add the results; the sum is the y-coordinate:

$$1920 + 80 = 2000$$

Look in Section 6 for a sample subroutine for decoding 4010 xy-reports.

COMPLEX VARIATIONS OF REPORT PARAMETERS

Complex variations of report parameters consist of a sequence of basic report parameters, just as complex host parameters consist of several basic command parameters. Since the complex report parameters have the same construction as the complex command parameters, we've included only brief descriptions of each complex report parameter here. Refer to *Host Parameters* in the first part of this section for further details and examples of construction.

String Report Parameters

The *string report parameter* consists of ASCII characters preceded by an integer report that contains the number of characters in the string. If there are no characters in the string (an empty string), the count is 0.

Integer Array Report Parameters

An *integer array report parameter* consists of a series of integer reports. The first character(s) in an integer array report is the *count*, an integer report that gives the number of individual array items to follow. If there are no integer reports in the array (an empty array), the count is 0.

THE EOL STRING

The *EOL string* is a string of no more than two ASCII characters (assigned in the most recent SET EOL STRING command). The terminal uses this string to mark the end of a line of data reported to the host and to break up long reports. Typically this string consists of the single character, `CR`. The EOL string in reports allows the terminal to pace the flow of data to the host (see Section 2 for an explanation of Prompt mode).

The routines you develop to parse reports must include parsing for EOL strings, even though the terminal rarely sends them. The terminal may insert an EOL string in a long complex report, but only if there is no other way to avoid exceeding the current maximum line length.

This discussion explains how the EOL string is used in reports.

The terminal sends a *final EOL string* at the end of all 4100-style reports except the 4010 Status Report. This final EOL string signals the end of the report and helps to ensure that the host application program receives the preceding characters in a timely manner. (In many host operating systems, the application program does not actually receive a message from the terminal until the message ends with a `CR`.)

The terminal also sends an EOL string to break up any report longer than 72 characters and to indicate the individual parts of the Error Report.

The terminal does not send the EOL string as part of ANSI reports, the answerback string, or the 4010 Status Report.

REPORT DESCRIPTIONS

The terminal uses the reports described here to return graphics or terminal status data to the host. When the terminal sends any of these reports to the host, it automatically enters Bypass mode. Refer to the discussion on Bypass mode in Section 2, and the ENTER BYPASS MODE command earlier in this section.

To describe the reports in this manual, we've listed each part of the report, then individually described each unique part of the report. Wherever possible, the names given to parts of reports are the same names used for command parameters. We've formatted these descriptions in a manner similar to the one used to describe the command parameters in the command descriptions in this section.

The Answerback String

The *answerback string* is a string of up to twenty characters that the terminal sends to the host in response to an ENQUIRY command. You can use the Setup command SET ANSWERBACK STRING to define a unique answerback string for each terminal so the host application knows which terminal it is talking to and can control which data is available to a specific terminal.

The host can send the ENQUIRY command to ask the terminal to send the answerback string to identify itself. The ENQUIRY command will work in all host command modes. The terminal will not respond to an ENQUIRY command issued while in Local mode.

Error Report

The Error Report is sent in response to the REPORT ERRORS command. This report is actually a series of up to eight *reports-for-one-error*, followed by the EOL string.

The terminal sends a *report-for-one-error* for each of the eight most recently detected error codes. If fewer than eight errors have been detected since power-up or since the last REPORT ERRORS command, then the terminal sends fewer than eight *reports-for-one-error*. The Error Report has this format:

```
report-for-one-error . . .  
EOL string
```

(The three periods following the *report-for-one-error* parameter (. . .) indicate that that it can appear more than once in the report.)

Each *report-for-one-error* describes an error in the following format:

```
error-code  
severity-level  
error-count  
EOL string
```

The following paragraphs detail the parts of the *report-for-one-error* in an Error Report.

error-code: four character-reports; consists of the opcode (two characters), the number of the parameter's position in the command causing the error, followed by an error-type digit. Refer to Appendix B for an explanation and list of error codes.

severity-level: integer report; specifies the severity level of the error that occurred; see Appendix B for an explanation of severity levels.

error-count: integer report; reports the number of times the terminal has detected that error since power-up or since the last REPORT ERRORS command.

After the last *report-for-one-error*, the terminal sends an EOL string.

Terminal Settings Report

The terminal sends the Terminal Settings Report in response to the REPORT TERMINAL SETTINGS command. The report has the following format:

```
opcode-report  
parameter-report ...  
EOL string
```

The following paragraphs describe the parts of a Terminal Settings Report.

opcode-report: two character-reports; returns either the opcode for the commands or one of the special inquiry codes listed in Table 5-8.

parameter-report: report parameter type depends on query; returns the command parameter values for the command specified in the *opcode-report* in the order that they appear in the command.

The two characters reported in the *opcode report* are the same two characters used in the REPORT TERMINAL SETTINGS command. However, if the REPORT TERMINAL SETTINGS command specifies an opcode for a command that does not exist in the terminal, the *opcode-report* is **SPSP**.

The type of *parameter-report* depends on the the type of parameter value being reported. For instance, the SET BAUD RATES command has two integer report parameters. Therefore, the Terminal Settings Report for SET BAUD RATES has two integer reports.

For special inquiry codes, use the *parameter-reports* listed in Table 5-8.

Table 5-8
SPECIAL INQUIRY CODES

Code	Parameter Reports
?M	integer report: <i>available-memory</i> integer report: <i>largest-contiguous-block</i> (The available program memory, and the size of the largest contiguous block of program memory, are reported as a number of 16-byte units of memory.)
?T	integer report: <i>model-number-code</i>
00	integer report: <i>standard-firmware-version-number</i>

Examples of Terminal Settings Report

Reporting Baud Rates. The REPORT TERMINAL SETTINGS command for the SET BAUD RATES command (opcode *NR*) queries the terminal for a report of its current baud rate settings.

Assume that the current EOL string is the single character, C_R , and that the terminal is set to transmit and receive at 1200 baud. In this case, the report that the terminal would send to the host is:

NR! + 0! + 0 C_R

Since the SET BAUD RATES command has two parameters, *transmit-data-rate* and *receive data rate*, these are the two *parameter-reports* returned in the REPORT TERMINAL SETTINGS command. Broken down, here's what each character means:

NR

The report is for opcode SET BAUD RATES (the *opcode-report*).

! + 0

The transmit rate is set to 1200 (first *parameter-report*).

! + 0

The receive rate is set to 1200 (second *parameter-report*).

C_R

The EOL string is set to C_R .

Reporting the Amount of Program Memory. To request a report on the amount of program memory available, the host sends a REPORT TERMINAL SETTINGS command with the inquiry code *?M*. If the EOL string is the same character as in the previous example, and the dialog buffer is set to 200, then the report that the terminal would send the host is:

?M !4 : C_R

The two *parameter-reports* are the amount of available program memory and the largest block, which are represented as integer reports. Broken down, here's what each character means:

?M

Report is in response to special inquiry code *?M* (the *opcode-report*).

!4

There are 20 16-byte blocks of program memory available (first *parameter-report*).

:

The largest contiguous block of program memory is 10 16-byte blocks (second *parameter-report*).

C_R

The EOL string is C_R .

Exceptions for Terminal Settings Report

For two commands, the meanings of the parameters reported in the Terminal Settings Report differ from the meanings of the command parameters sent to the terminal. These commands are:

PROMPT MODE (opcode *NM*)
SET SURFACE COLOR MAP (opcode *TG*)

PROMPT MODE. Prompt mode can be turned on with a parameter of 1 or 2. However, the terminal reports 1 if Prompt mode is on and 2 if Prompt mode is off.

SET SURFACE COLOR MAP. A Terminal Settings Report for this command has two *parameter-reports*: an integer report for *number-of-surfaces*, and an integer array report for *color-mixtures-array*.

The integer report for *number-of-surfaces* is the number of surfaces currently defined. Since the 4105 terminal can display just one surface, this integer report will always be 1. (This report is included for compatibility with other Tektronix terminals that can display more than one surface).

The *color-mixtures-array* parameter report contains the following:

- The background color mixture values
- The surface number
- The color mixture values for each of the color indices

The background color mixture values are given in three integer reports.

The terminal sends the surface number as a negative value in an integer report, followed by an integer report triplet for each color index assigned to that surface. Each triplet contains the color mixture values for the index. The color indices are reported in numerical order following the surface number.

Here is an example of the information contained in a *color-mixtures-array* integer array report, which is a long series of integer reports (spaces have been added here to break the report up for readability; Space characters that are actually part of the report are indicated explicitly as *Sp*):

```
TG SpSp1 Sp!9 SpSp0SpSp0SpSp0 SpSp! SpSp0 Sp&4SpSp0  
Sp!8Sp#2Sp&4 Sp/0Sp#2Sp&4 SpSp0Sp #2Sp&4  
Sp2<Sp#2Sp&4 Sp#<Sp#2Sp&4 Sp + 4Sp#2Sp&4 CR
```

Broken down, here's what each parameter in the report means:

TG

This report is for the SET SURFACE COLOR MAP command (the *opcode-report*).

SpSp1

There is one surface (first *parameter-report*).

Sp!9

Here, the first integer report says that there are 25 integer reports to follow in the integer array report (part of second *parameter-report*).

SpSp0SpSp0SpSp0

These three integer reports say that the background color is black — HLS coordinates 0,0,0 (part of second *parameter-report*).

SpSp!

This integer report is the integer -1, which means the report is for Surface 1 (part of second *parameter-report*).

SpSp0Sp&4SpSp0

These three integer reports say that Index 1 is set to *white* with HLS coordinates 0,100,0 (part of second parameter report).

$Sp\ 1\ 8Sp\#2Sp\&4$

These three integer reports say that Index 2 is set to *red* with HLS coordinates 120,50,100 (part of second *parameter-report*).

$Sp/0Sp\#2Sp\&4$

These three integer reports say that Index 3 is set to *green* with HLS coordinates 240,50,100 (part of second *parameter-report*).

$SpSp0Sp\#2SpSp\&4$

These three integer reports say that Index 4 is set to *blue* HLS coordinates 0,50,100 (part of second *parameter-report*).

$Sp2<Sp\#2Sp\&4$

These three integer reports say that Index 5 is set to *cyan* HLS coordinates 300,50,100 (part of second *parameter-report*).

$Sp\#\<Sp\#2Sp\&4$

These three integer reports say that Index 6 is set to *magenta* HLS coordinates 60,50,100 (part of second *parameter-report*).

$Sp\ +\ 4Sp\#2Sp\&4$

These three integer reports say that Index 7 is set to *yellow* HLS coordinates 180,50,100 (part of second *parameter-report*).

C_R

The EOL string is C_R .

4010 GIN Report

When 4010 GIN is enabled and the user presses a key to send the cursor position to the host program, the terminal generates a 4010 GIN Report. This report tells the host program which key the user pressed and the position of the GIN cursor in terminal space.

NOTE

The 4010 GIN Report regards terminal space as a 1024x1024 area rather than the 4096x4096 area used when specifying locations for display. Reported coordinate values must be multiplied by 4 to give coordinates consistent with those used in other commands.

If the user presses a key that has a key macro defined for it, the 4010 GIN Report is sent, but the key-character sent in the report is the first character intended for the host in the macro definition. The remainder of the macro is expanded normally.

The 4010 GIN Report has the following format:

key
cursor-position
 EOL string

key: character report; specifies the ASCII key that the user pressed.

cursor-position: 4010 xy-report; reports the location of the graphics cursor.

Since only the ten most significant bits of the x- and y-coordinates are reported, the reported values are an approximation of the graphics cursor position. Section 6 contains a sample FORTRAN routine that decodes the 4010 GIN Report.

4010 Status Report

This report is sent in response to REPORT 4010 STATUS. The report has two forms, depending on whether 4010 GIN is enabled when the command is sent.

If 4010 GIN is *not* enabled, the report has the following format:

terminal-status
alpha-cursor-position
EOL string

If 4010 is enabled, the report has the following format:

graphics-cursor-position
EOL string

terminal-status: character report; reports the terminal status encoded into the seven bits of an ASCII character, shown in Table 5-9.

Bits 7 and 6 are always set to 0 and 1, respectively; Bits 2 and 1 are also set to 0 and 1, respectively.

The HCU (Bit 5) is set to 0 if a copier is attached to the COPIER port and is ready to accept a copy request; otherwise this bit is set to 1.

Bits 4 and 3 indicate the terminal's implicit command mode status as shown in Table 5-10.

For example, if the terminal (1) has a hard copy unit attached, (2) is ready for a hardcopy command, and (3) is in Vector mode, the bits sent are:

0101001

The corresponding character on the ASCII chart is the closing parenthesis —) — which would be transmitted as the status byte.

alpha-cursor-position and *graphics-cursor-position*: 4010 xy-report; reports in 10-bit form the position of either the alpha cursor or the graphics cursor.

Look in Section 6 for the sample FORTRAN routine that decodes 4010 xy-reports.

Table 5-9

TERMINAL STATUS CHARACTER BITS

B7	B6	B5	B4	B3	B2	B1
0	1	HCU	V	A	0	1

Table 5-10

IMPLICIT COMMAND MODE STATUS

V	A	Mode Status
0	0	The terminal is in Marker mode
0	1	The terminal is in Alpha mode
1	0	The terminal is in Vector mode
1	1	This combination doesn't occur

Section 6

PROGRAMMING EXAMPLES

This section contains subroutines that encode terminal commands and decode terminal reports, and a program example that uses the subroutines to perform a simple graphics task. The subroutines we've included are:

- An initialization routine that must be called prior to using any of the other routines
- A group of subroutines for encoding command parameters
- A group of subroutines for decoding report parameters
- Specifications for the low-level I/O support subroutines

These subroutines (except for the low-level I/O support subroutines) and the program example are written in standard FORTRAN 77. The low-level I/O support routines cannot be written in standard FORTRAN-77, but — by using nonstandard techniques — you can write them on most commercially available FORTRAN compilers.

NOTE

The subroutines and program example in this section work in the CP/M-86 operating system as used on the Tektronix 4170 Local Graphics Processing Unit. They are included here to show one way to program for the 4105 terminal. This code may not run as written in any other environment.

PROGRAMMING EXAMPLES

INITIALIZATION ROUTINE

```
      SUBROUTINE Init (jterm)

C *****
C *
C *   Initializes terminal communications parameters for use by the
C *   following routines. Gets the terminal type. Initializes
C *   common variables.
C *
C * Parameters
C * Output:
C *   jterm   - terminal id (for example, 4105)
C *
C * Global Area /Grphcs/:
C *   kterm   - terminal id
C *   kchars  - used to optimize XY coordinate output
C *   lfirst  - .TRUE. if no XY coordinate has been output
C *
C *****

      COMMON /Grphcs/ kterm,kchars(5),lfirst
      CHARACTER*10 repstr
      LOGICAL lfirst

C Initialize the communications parameters so terminal and GIN
C reports can be done. The subroutines which follow assume that
C these values will not be changed.

C Set terminal mode to code Tek
      CALL Cmd1x ('%!')
      CALL Putchr (48)

C Set End-Of-Message characters to <CR><NL>
      CALL Cmd1x ('NC')
      CALL Int1x (13)
      CALL Int1x (0)

C Set End-Of-Line string to <CR>
      CALL Cmd1x ('NT')
      CALL Inry1x (1,13)

C Set bypass cancel character to <LF>
      CALL Cmd1x ('NU')
      CALL Int1x (10)
```

```

C Request terminal id, read and decode it; set global kterm
  CALL Cmd1x ('IQ')
  CALL Putchr (63)
  CALL Putchr (84)
  CALL Gtrp (10,repstr,jgot)
  CALL Gtin (repstr,3,jterm,jnext)
  kterm = jterm
C Initialize the optimized graphics characters
  DO 100 i=1,5
    kchars(i) = -1
100  CONTINUE

C No XY coordinates have been output yet, so set lfirst appropriately
  lfirst = .TRUE.

  RETURN
  END

```

COMMAND PARAMETER ENCODING SUBROUTINES

```

          SUBROUTINE Cmd1x (opcode)

C *****
C *
C * Output a two-character terminal command sequence.
C *
C * Parameters
C * Input:
C *   opcode - character string containing the two-character
C *           sequence
C *
C *****

          CHARACTER*(*) opcode

C Precede the two characters with an <EC>
          CALL Putchr (27)

C Output the two characters
          ichr = ICHAR (opcode(1:1))
          CALL Putchr (ichr)
          ichr = ICHAR (opcode(2:2))
          CALL Putchr (ichr)

          RETURN
          END

```

PROGRAMMING EXAMPLES

```

SUBROUTINE Int1x (intarg)
C *****
C *
C * Encode an integer parameter and output it.
C *
C * Parameters
C * Input:
C *   intarg - integer value to encode and output
C *
C *****

C Break the integer value into three encoded bytes
      jint=IABS(intarg)
      jhi1=jint/1024+64
      jhi2=MOD(jint/16,64)+64
      jloi=MOD(jint,16)+32
      IF (intarg .GE. 0) jloi=jloi+16

C Decide whether to output 1, 2 or 3 bytes
      IF (jhi1 .NE. 64) GO TO 10
      IF (jhi2 .NE. 64) GO TO 20
      GO TO 30

10      CALL Putchr (jhi1)
20      CALL Putchr (jhi2)
30      CALL Putchr (jloi)

      RETURN
      END

```

```

SUBROUTINE Inry1x (len,inry)
C *****
C *
C * Encode and output an integer array.
C *
C * Parameters
C * Input:
C *   len      - array length
C *   inry     - array to encode and output
C *
C *****

      INTEGER inry(*)

C Output the array length
      CALL Int1x (len)

C Output each of the array elements
      IF (len .LE. 0) GO TO 999
      DO 100 i=1,len
         CALL Int1x (inry(i))
100     CONTINUE

999     RETURN
      END

```



```
          SUBROUTINE Str1x (len,string)
C *****
C *
C * Encode and output a character array.
C *
C * Parameters
C *   Input:
C *     len      - string length
C *     string   - character string to be encoded and output
C *
C *****

          CHARACTER*(*) string

C Output the array length
          CALL Int1x (len)

C Output each of the characters
          IF (len .LE. 0) goto 999
          DO 100 i=1,len
             ichr = ICHAR(string(i:i))
             CALL Putchr (ichr)
100      CONTINUE
999      RETURN
          END
```

PROGRAMMING EXAMPLES

```
      SUBROUTINE Xy1x (ix,iy)
C *****
C *
C * Encode and output an XY parameter.
C *
C * Parameters
C * Input:
C *   ix, iy - XY coordinate to encode and output
C *
C *****

      COMMON /Grphcs/ kterm,kchars(5),lfirst
      LOGICAL lfirst

C Calculate all five encoded bytes
      khiy=iy/128+32
      keb=MOD(iy,4)*4+MOD(ix,4)+96
      kloy=MOD(iy/4,32)+96
      khix=ix/128+32
      klox=MOD(ix/4,32)+64

C Output encode bytes as needed
      IF (lfirst .OR. khiy.NE.kchars(1)) CALL Putchr (khiy)
      IF (lfirst .OR. keb.NE.kchars(2)) CALL Putchr (keb)
      IF (lfirst .OR. keb.NE.kchars(2) .OR. kloy.NE.kchars(3) .OR.
&      khix.NE.kchars(4)) CALL Putchr (kloy)
      IF (lfirst .OR. khix.NE.kchars(4)) CALL Putchr (khix)
      CALL Putchr (klox)

C Update optimization array
      kchars(1)=khiy
      kchars(2)=keb
      kchars(3)=kloy
      kchars(4)=khix
      kchars(5)=klox
      lfirst = .FALSE.

      RETURN
      END
```

TERMINAL REPORT DECODING SUBROUTINES

```

SUBROUTINE Gtrp(imaxin,repstr,jgot)
C *****
C *
C * Get a terminal report. This routine provides the base on which
C * all the following routines operate.
C *
C * Parameters
C * Input:
C *   imaxin - maximum report length which will be accepted;
C *           additional characters received are discarded
C * Output:
C *   repstr - string containing the report
C *   jgot   - number of characters in the report
C *
C *****

CHARACTER repstr*(*)

jgot = 0

C Use the special Getchr routine to get reports. Using standard
C FORTRAN it is possible to read a report string (if the End-Of-Line
C character is <<CR>>), but not to get its correct length since
C FORTRAN will pad the input string with blanks.

100 CALL Getchr (ichr)

C A <CR> signals the end of the report.
IF (ichr .EQ. 13) GO TO 110

C Add the character to the string if there is enough room, else
C discard the character.
IF (jgot .LT. imaxin) THEN
    jgot = jgot + 1
    repstr(jgot:jgot) = CHAR(ichr)
ENDIF
GO TO 100

C Output a <LF> to cancel bypass mode.
110 CALL Putchr (10)

RETURN
END

```

PROGRAMMING EXAMPLES

```
          SUBROUTINE Gt10 (keychr,jx,jy)
C *****
C *
C * Read and decode a 4010 GIN report. The routine will wait until
C * a 4010 GIN report is available.
C *
C * Parameters
C * Output:
C *   keychr - character typed by the user to create the 4010
C *           GIN report
C *   Jx, jy - 4010 GIN position
C *****

          COMMON /Grphcs/ kterm,kchars(5),lfirst
          CHARACTER*1 keychr
          CHARACTER*10 cginbf

C Use the general-purpose report routine to read the 4010 GIN report
          CALL Gtrp (10,cginbf,kgot)

C The first character in the report is the key struck
          keychr = cginbf(1:1)

C Characters 2-5 are the 4010 GIN report. Call the decoding routine.
          CALL Xy10 (cginbf,2,jx,jy,jnext)

          RETURN
          END
```

```

SUBROUTINE Xy10 (repstr,istart,jx,jy,jnext)
C *****
C *
C * Decode a 4010 XY report.
C *
C * Parameters
C * Input:
C *   repstr - report string (obtained from Gtrp)
C *   istart - starting character position of the 4010 XY report
C *           in the report string
C * Output:
C *   jx, jy - decoded XY coordinate
C *   jnext - starting position of next encoded parameter in
C *           the report string
C *****

CHARACTER*(*) repstr
INTEGER*2 hiy,loy,hix,lox

C The XY coordinate is always encoded in 4 bytes.
  jnext = istart
  hix=ICHAR(repstr(jnext:jnext))
  jnext=jnext+1
  lox=ICHAR(repstr(jnext:jnext))
  jnext=jnext+1
  hiy=ICHAR(repstr(jnext:jnext))
  jnext=jnext+1
  loy=ICHAR(repstr(jnext:jnext))
  jnext=jnext+1

C Convert the encoded bytes into separate X and Y
  jx=MOD(hix,32)
  jx=(jx*32+MOD(lox,32))*4
  jy=MOD(hiy,32)
  jy=(jy*32+MOD(loy,32))*4

RETURN
END

```

PROGRAMMING EXAMPLES

```

SUBROUTINE Gtch (repstr,istart,jlen,jchstr,jnext)
C *****
C *
C * Decode a character string report.
C *
C * Parameters
C * Input:
C *   repstr - report string (obtained from Gtrp)
C *   istart - starting character position of the character
C *           string report in the report string
C * Output:
C *   jlen   - length of the obtained character string
C *   jchstr - character string
C *   jnext  - starting position of next encoded parameter in
C *           the report string
C *
C *****
CHARACTER*(*) repstr,jchstr

C Get the character string report length
  CALL Gtin (repstr,istart,jlen,jnext)

C Get the character string itself
  jchstr = repstr(jnext:jnext+jlen-1)
  jnext = jnext + jlen

  RETURN
  END
```

```

SUBROUTINE Gtin (repstr,istart,jval,jnext)
C *****
C *
C * Decode an integer report.
C *
C * Parameters
C * Input:
C *   repstr - report string (obtained from Gtrp)
C *   istart - starting character position of the integer
C *           report in the report string
C * Output:
C *   jval   - decoded integer value
C *   jnext  - starting position of next encoded parameter in
C *           the report string
C *
C *****

CHARACTER*(*) repstr
INTEGER*2 hii1,hii2,loi

C An integer report is always encoded in 3 bytes
  jnext = istart
  hii1 = ICHAR(repstr(jnext:jnext)) - 32
  jnext = jnext + 1
  hii2 = ICHAR(repstr(jnext:jnext)) - 32
  jnext = jnext + 1
  loi = ICHAR(repstr(jnext:jnext)) - 32
  jnext = jnext + 1

C Decode the 3 bytes to obtain the integer value
  jval = hii1*1024 + hii2*16 + MOD(loi,16)
  IF (MOD(loi/16,2) .EQ. 0) jval = -jval

RETURN
END

```

LOW-LEVEL I/O SUPPORT SUBROUTINES

SUBROUTINE Getchr (ichr)

```
C *****  
C *  
C * Get a character from the terminal. This subroutine cannot be  
C * coded in standard FORTRAN-77 and must be customized for  
C * each FORTRAN environment.  
C *  
C * Parameters  
C * Output:  
C *   ichr   - ASCII decimal equivalent of the character read  
C *           (e.g. 'A' = 65).  
C *  
C *****  
  
C **** Customized code goes here ****  
  
      RETURN  
      END
```

SUBROUTINE Putchr (ichr)

```
C *****  
C *  
C * Send a character to the terminal. This subroutine cannot be  
C * coded in standard FORTRAN-77 and must be customized for  
C * each FORTRAN environment.  
C *  
C * Parameters  
C * Output:  
C *   ichr   - ASCII decimal equivalent of the character to  
C *           write  
C *  
C *****  
  
C **** Customized code goes here ****  
  
      RETURN  
      END
```


SAMPLE PROGRAM

```

C *****
C *
C * This is a sample main program which uses some of the subroutines
C * listed above. The program lets the user draw lines in the
C * graphics area.
C *
C *****

CHARACTER*1 sigchr, keychr

C Initialize the graphic subroutines
CALL Init (jterm)

C Disable the dialog area and make it invisible
CALL Cmd1x ('KA')
CALL Int1x (0)
CALL Cmd1x ('LV')
CALL Int1x (0)

C Print instructions for use
CALL Cmd1x ('LF')
CALL Xy1x (0,100)
print *,'Enter "D" to draw, "M" to move, "S" to end'

C Set old beam position
ix = 0
iy = 0

C Enable 4010 GIN (<EC><SB>)
100 CALL Putchr (27)
CALL Putchr (26)

C Get a point
CALL Gt10 (keychr,jx,jy)

C Move to old beam position and save new position
CALL Cmd1x ('LF')
CALL Xy1x (ix,iy)
ix = jx
iy = jy

C Decide what to do with the point
IF (keychr .EQ. 'S' .OR. keychr .EQ. 's') GO TO 900

IF (keychr .EQ. 'D' .OR. keychr .EQ. 'd') THEN
CALL Cmd1x ('LG')
CALL Xy1x (jx,jy)
ELSE IF (keychr .EQ. 'M' .OR. keychr .EQ. 'm') THEN
CALL Cmd1x ('LF')
CALL Xy1x (jx,jy)
ENDIF

GO TO 100

C Enable the dialog area and make it visible
900 CALL Cmd1x ('KA')
CALL Int1x (1)
CALL Cmd1x ('LV')
CALL Int1x (1)

END

```


Appendix A

CODE CHARTS AND KEYBOARD MACROS

There are eight different character sets available on your terminal. Six character sets support language-dependent keyboards; two character sets provide supplementary symbols and rulings symbols. You can use any of these character sets from the keyboard supplied with your terminal.

When you power up the terminal, it automatically selects the appropriate character set for your keyboard and uses that character set as the default character set. For example, the North American keyboard selects the ASCII character set, while the Option 4G German keyboard selects the German character set.

You are not limited to the character set corresponding to your keyboard; you can select any two of the eight available character sets and switch back and forth between them. Enter ANSI mode and use the SCS (SELECT CHARACTER SET) command to assign different character sets as the G0 and G1 character sets. Then you can use the ANSI SI (SHIFT IN) and SO (SHIFT OUT) commands — or the TEK mode SET ALPHATEXT FONT command — to switch between the two character sets.

For each language-dependent character set, this appendix contains:

- A code chart
- A keyboard macro chart

The macro charts list the macro numbers invoked by each key and key combination available on a specific keyboard. The code charts contain the binary and ADE values for the alphanumeric characters and control characters in the character set.

You can find more information about macros and how to define and use them under *Using Macros* in Section 4. Look for specific information about the DEFINE MACRO, DEFINE NONVOLATILE MACRO, LEARN, and LEARN NONVOLATILE commands in Section 5.

The code charts for the Supplementary and Rulings character sets are at the end of this appendix.

ASCII/NORTH AMERICAN CHARACTER SET

BITS				0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
B7	B6	B5		CONTROL		FIGURES		UPPERCASE		LOWERCASE	
B4	B3	B2	B1								
0	0	0	0	NU ₀	DL ₁₆	Sp ₃₂	0 ₄₈	@ ₆₄	P ₈₀	\ ₉₆	p ₁₁₂
0	0	0	1	SH ₁	D1 ₁₇	! ₃₃	1 ₄₉	A ₆₅	Q ₈₁	a ₉₇	q ₁₁₃
0	0	1	0	SX ₂	D2 ₁₈	" ₃₄	2 ₅₀	B ₆₆	R ₈₂	b ₉₈	r ₁₁₄
0	0	1	1	EX ₃	D3 ₁₉	# ₃₅	3 ₅₁	C ₆₇	S ₈₃	c ₉₉	s ₁₁₅
0	1	0	0	ET ₄	D4 ₂₀	\$ ₃₆	4 ₅₂	D ₆₈	T ₈₄	d ₁₀₀	t ₁₁₆
0	1	0	1	EQ ₅	NK ₂₁	% ₃₇	5 ₅₃	E ₆₉	U ₈₅	e ₁₀₁	u ₁₁₇
0	1	1	0	AK ₆	SY ₂₂	& ₃₈	6 ₅₄	F ₇₀	V ₈₆	f ₁₀₂	v ₁₁₈
0	1	1	1	BL ₇	EB ₂₃	/ ₃₉	7 ₅₅	G ₇₁	W ₈₇	g ₁₀₃	w ₁₁₉
1	0	0	0	BS ₈	CN ₂₄	(₄₀	8 ₅₆	H ₇₂	X ₈₈	h ₁₀₄	x ₁₂₀
1	0	0	1	HT ₉	EM ₂₅) ₄₁	9 ₅₇	I ₇₃	Y ₈₉	i ₁₀₅	y ₁₂₁
1	0	1	0	LF ₁₀	SB ₂₆	* ₄₂	: ₅₈	J ₇₄	Z ₉₀	j ₁₀₆	z ₁₂₂
1	0	1	1	VT ₁₁	EC ₂₇	+ ₄₃	; ₅₉	K ₇₅	[₉₁	k ₁₀₇	{ ₁₂₃
1	1	0	0	FF ₁₂	FS ₂₈	, ₄₄	< ₆₀	L ₇₆	\ ₉₂	l ₁₀₈	₁₂₄
1	1	0	1	CR ₁₃	GS ₂₉	- ₄₅	= ₆₁	M ₇₇] ₉₃	m ₁₀₉	} ₁₂₅
1	1	1	0	SO ₁₄	RS ₃₀	. ₄₆	> ₆₂	N ₇₈	^ ₉₄	n ₁₁₀	~ ₁₂₆
1	1	1	1	SI ₁₅	US ₃₁	/ ₄₇	? ₆₃	O ₇₉	_ ₉₅	o ₁₁₁	DT ₁₂₇

(4526)4893-18

Figure A-1. ASCII/North American Character Set Code Chart.


										JOYDISK				G Erase Cancel D Copy Menu Dialog Setup S Copy				F1 F2 F3 F4				F5 F6 F7 F8								
										RIGHT	UP	LEFT	DOWN																	
										Unshifted	-135	-136	-137	-138	-111	-112	-113	-114	128	129	130	131	132	133	134	135				
										Shifted	-139	-140	-141	-142	-117	-118	-119	-120	136	137	138	139	140	141	142	143				
										Ctrl	-143	-144	-145	-146	-123	-124	-125	-126	-2	-3	-4	-5	-6	-7	-8	-9				
										Ctrl-Shifted	-147	-148	-149	-150	-129	-130	-131	-132	-10	-11	-12	-13	-14	-15	-16	-17				
										D Erase {			@	#	\$	%	^	&	*	()	-	+	}	Rub Out					
										S Erase [2	3	4	5	6	7	8	9	0	-	=]						
										Unshifted	-115	91	49	50	51	52	53	54	55	56	57	48	45	61	93	127				
										Shifted	-121	123	33	64	35	36	37	94	38	42	40	41	95	43	125	-34				
										Ctrl	-127	27	49	50	51	52	53	54	55	56	57	48	45	61	29	-35				
										Ctrl-Shifted	-133	27	33	0	35	36	37	30	38	42	40	41	31	43	29	-36				
										Esc		~	Q	W	E	R	T	Y	U	I	O	P	\	Back Space	Line Feed					
										Unshifted	27	124	113	119	101	114	116	121	117	105	111	112	92	8	10					
										Shifted	-37	126	81	87	69	82	84	89	85	73	79	80	96	-40	-43					
										Ctrl	-38	124	17	23	5	18	20	25	21	9	15	16	28	-41	-44					
										Ctrl-Shifted	-39	126	17	23	5	18	20	25	21	9	15	16	28	-42	-45					
										Tab		Ctrl	A	S	D	F	G	H	J	K	L	:	"/	Return						
										Unshifted	9		97	115	100	102	103	104	106	107	108	59	39	13						
										Shifted	-46		65	83	68	70	71	72	74	75	76	58	34	-49						
										Ctrl	-47		1	19	4	6	7	8	10	11	12	59	39	-50						
										Ctrl-Shifted	-48		1	19	4	6	7	8	10	11	12	58	34	-51						
										Caps Lock		Shift	Z	X	C	V	B	N	M	<	>	?	Shift	Break						
										Unshifted	122	120	99	118	98	110	109	44	46	47				-116						
										Shifted	90	88	67	86	66	78	77	60	62	63				-122						
										Ctrl	26	24	3	22	2	14	13	44	46	47				-128						
										Ctrl-Shifted	26	24	3	22	2	14	13	60	62	63				-134						
																				SPACEBAR										
																				Unshifted	32									
																				Shifted	-52									
																				Ctrl	-53									
																				Ctrl-Shifted	-54									
																						-68	-82	-96	-110					
																						-55	-65	-79	-93					
																						-69	-79	-93	-107					
																						-83	-93	-107						

Figure A-2. ASCII/North American Keyboard Macro Chart.

4893-17

UNITED KINGDOM CHARACTER SET

BITS				0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
B7	B6	B5		CONTROL		FIGURES		UPPERCASE		LOWERCASE	
B4	B3	B2	B1								
0	0	0	0	NU ₀	DL ₁₆	Sp ₃₂	0 ₄₈	@ ₆₄	P ₈₀	\ ₉₆	p ₁₁₂
0	0	0	1	SH ₁	D1 ₁₇	! ₃₃	1 ₄₉	A ₆₅	Q ₈₁	a ₉₇	q ₁₁₃
0	0	1	0	SX ₂	D2 ₁₈	" ₃₄	2 ₅₀	B ₆₆	R ₈₂	b ₉₈	r ₁₁₄
0	0	1	1	EX ₃	D3 ₁₉	£ ₃₅	3 ₅₁	C ₆₇	S ₈₃	c ₉₉	s ₁₁₅
0	1	0	0	ET ₄	D4 ₂₀	\$ ₃₆	4 ₅₂	D ₆₈	T ₈₄	d ₁₀₀	t ₁₁₆
0	1	0	1	EQ ₅	NK ₂₁	% ₃₇	5 ₅₃	E ₆₉	U ₈₅	e ₁₀₁	u ₁₁₇
0	1	1	0	AK ₆	SY ₂₂	& ₃₈	6 ₅₄	F ₇₀	V ₈₆	f ₁₀₂	v ₁₁₈
0	1	1	1	BL ₇	EB ₂₃	/ ₃₉	7 ₅₅	G ₇₁	W ₈₇	g ₁₀₃	w ₁₁₉
1	0	0	0	BS ₈	CN ₂₄	(₄₀	8 ₅₆	H ₇₂	X ₈₈	h ₁₀₄	x ₁₂₀
1	0	0	1	HT ₉	EM ₂₅) ₄₁	9 ₅₇	I ₇₃	Y ₈₉	i ₁₀₅	y ₁₂₁
1	0	1	0	LF ₁₀	SB ₂₆	* ₄₂	: ₅₈	J ₇₄	Z ₉₀	j ₁₀₆	z ₁₂₂
1	0	1	1	VT ₁₁	EC ₂₇	+ ₄₃	; ₅₉	K ₇₅	[₉₁	k ₁₀₇	{ ₁₂₃
1	1	0	0	FF ₁₂	FS ₂₈	, ₄₄	< ₆₀	L ₇₆	\ ₉₂	l ₁₀₈	₁₂₄
1	1	0	1	CR ₁₃	GS ₂₉	- ₄₅	= ₆₁	M ₇₇] ₉₃	m ₁₀₉	} ₁₂₅
1	1	1	0	SO ₁₄	RS ₃₀	. ₄₆	> ₆₂	N ₇₈	^ ₉₄	n ₁₁₀	~ ₁₂₆
1	1	1	1	SI ₁₅	US ₃₁	/ ₄₇	? ₆₃	O ₇₉	_ ₉₅	o ₁₁₁	DT ₁₂₇

(4526)4893-20B

Figure A-3. United Kingdom Character Set Code Chart.

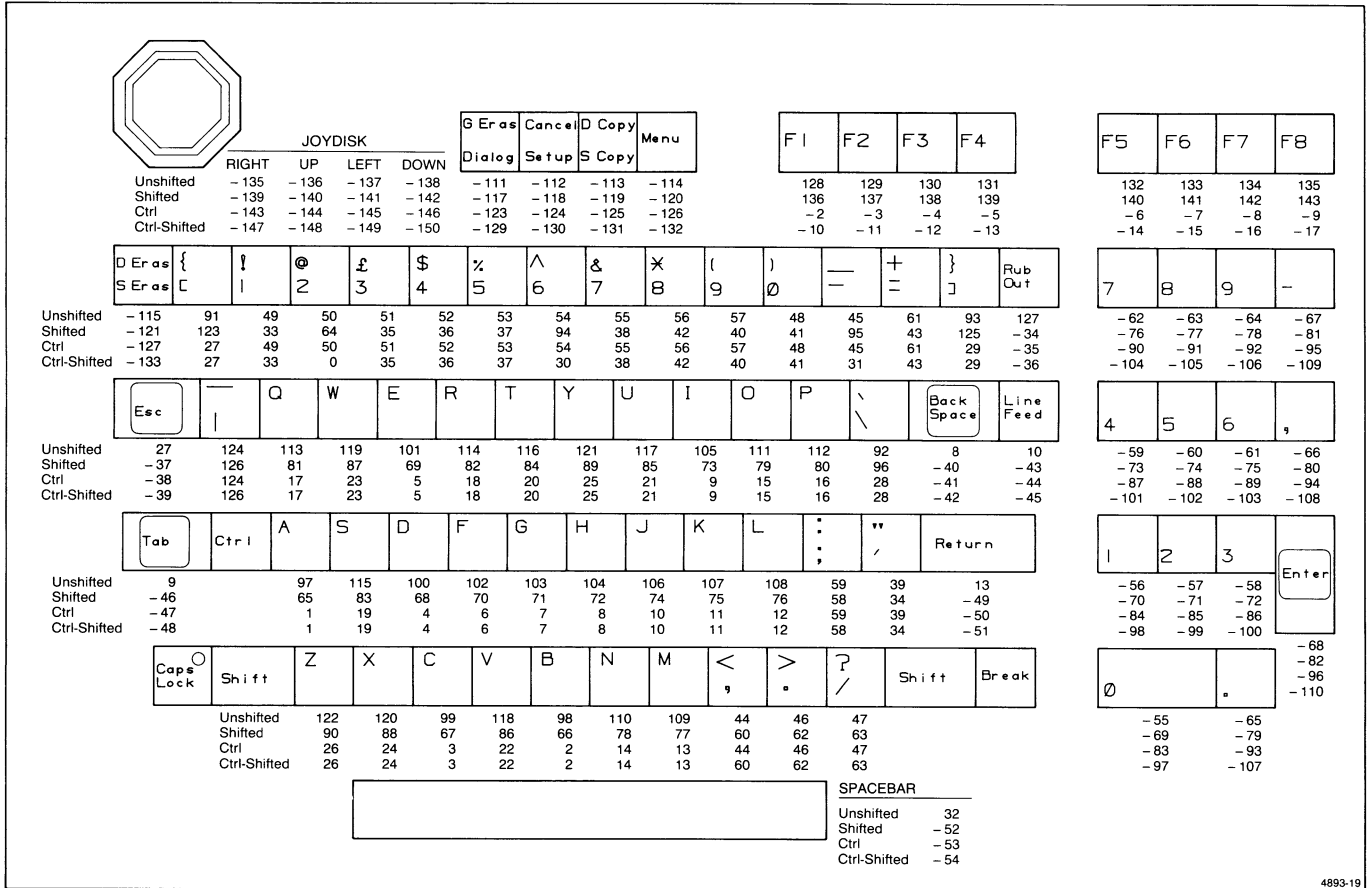


Figure A-4. United Kingdom Keyboard Macro Chart.


4893-19

FRENCH CHARACTER SET

BITS				0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
B7	B6	B5	B4	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
0	0	0	0								
0	0	0	0	NU ₀	DL ₁₆	Sp ₃₂	0 ₄₈	à ₆₄	P ₈₀	μ ₉₆	p ₁₁₂
0	0	0	1	SH ₁	D1 ₁₇	! ₃₃	1 ₄₉	A ₆₅	Q ₈₁	a ₉₇	q ₁₁₃
0	0	1	0	SX ₂	D2 ₁₈	" ₃₄	2 ₅₀	B ₆₆	R ₈₂	b ₉₈	r ₁₁₄
0	0	1	1	EX ₃	D3 ₁₉	£ ₃₅	3 ₅₁	C ₆₇	S ₈₃	c ₉₉	s ₁₁₅
0	1	0	0	ET ₄	D4 ₂₀	\$ ₃₆	4 ₅₂	D ₆₈	T ₈₄	d ₁₀₀	t ₁₁₆
0	1	0	1	EQ ₅	NK ₂₁	% ₃₇	5 ₅₃	E ₆₉	U ₈₅	e ₁₀₁	u ₁₁₇
0	1	1	0	AK ₆	SY ₂₂	& ₃₈	6 ₅₄	F ₇₀	V ₈₆	f ₁₀₂	v ₁₁₈
0	1	1	1	BL ₇	EB ₂₃	' ₃₉	7 ₅₅	G ₇₁	W ₈₇	g ₁₀₃	w ₁₁₉
1	0	0	0	BS ₈	CN ₂₄	(₄₀	8 ₅₆	H ₇₂	X ₈₈	h ₁₀₄	x ₁₂₀
1	0	0	1	HT ₉	EM ₂₅) ₄₁	9 ₅₇	I ₇₃	Y ₈₉	i ₁₀₅	y ₁₂₁
1	0	1	0	LF ₁₀	SB ₂₆	* ₄₂	: ₅₈	J ₇₄	Z ₉₀	j ₁₀₆	z ₁₂₂
1	0	1	1	VT ₁₁	EC ₂₇	+ ₄₃	; ₅₉	K ₇₅	° ₉₁	k ₁₀₇	é ₁₂₃
1	1	0	0	FF ₁₂	FS ₂₈	, ₄₄	< ₆₀	L ₇₆	ç ₉₂	l ₁₀₈	ù ₁₂₄
1	1	0	1	CR ₁₃	GS ₂₉	- ₄₅	= ₆₁	M ₇₇	§ ₉₃	m ₁₀₉	è ₁₂₅
1	1	1	0	SO ₁₄	RS ₃₀	. ₄₆	> ₆₂	N ₇₈	^ ₉₄	n ₁₁₀	" ₁₂₆
1	1	1	1	SI ₁₅	US ₃₁	/ ₄₇	? ₆₃	O ₇₉	— ₉₅	o ₁₁₁	DT ₁₂₇

(4526)4893-22B

Figure A-5. French Character Set Code Chart.

				JOYDISK				G Eras Cancel D Copy Dialog Setup S Copy Menu				F1 F2 F3 F4				F5 F6 F7 F8							
				RIGHT	UP	LEFT	DOWN																
Unshifted				-135	-136	-137	-138	-111	-112	-113	-114	128	129	130	131	132	133	134	135				
Shifted				-139	-140	-141	-142	-117	-118	-119	-120	136	137	138	139	140	141	142	143				
Ctrl				-143	-144	-145	-146	-123	-124	-125	-126	-2	-3	-4	-5	-6	-7	-8	-9				
Ctrl-Shifted				-147	-148	-149	-150	-129	-130	-131	-132	-10	-11	-12	-13	-14	-15	-16	-17				
D Eras X				I &	Z é	3 "	4 /	5 (6 §	7 è	8 !	9 ç	∅ ã	°)	-	µ £	⌫	7 8 9 -					
Unshifted				-115	36	38	123	34	39	40	93	125	33	92	64	41	45	35	127	-62	-63	-64	-67
Shifted				-121	42	49	50	51	52	53	54	55	56	57	48	91	95	96	-34	-76	-77	-78	-81
Ctrl				-127	36	38	27	34	39	40	29	29	33	28	0	41	45	35	-35	-90	-91	-92	-95
Ctrl-Shifted				-133	42	49	50	51	52	53	54	55	56	57	48	27	31	28	-36	-104	-105	-106	-109
Tab				>	A	Z	E	R	T	Y	U	I	O	P	..	Esc	←	4 5 6 ,					
Unshifted				9	60	97	122	101	114	116	121	117	105	111	112	94	27	8	-59	-60	-61	-66	
Shifted				-46	62	65	90	69	82	84	89	85	73	79	80	126	-37	-40	-73	-74	-75	-80	
Ctrl				-47	60	1	26	5	18	20	25	21	9	15	16	30	-38	-41	-87	-88	-89	-94	
Ctrl-Shifted				-48	62	1	26	5	18	20	25	21	9	15	16	126	-39	-42	-101	-102	-103	-108	
↓				Ctrl	Q	S	D	F	G	H	J	K	L	M	%	↵	1 2 3 Enter						
Unshifted				10	113	115	100	102	103	104	106	107	108	109	124	13	-56	-57	-58	-68			
Shifted				-43	81	83	68	70	71	72	74	75	76	77	37	-49	-70	-71	-72	-82			
Ctrl				-44	17	19	4	6	7	8	10	11	12	13	124	-50	-84	-85	-86	-96			
Ctrl-Shifted				-45	17	19	4	6	7	8	10	11	12	13	37	-51	-98	-99	-100	-110			
Caps Lock				MAJ m/n	W	X	C	V	B	N	? 9	° 0	/ :	+ =	MAJ m/n	Break	∅ .						
Unshifted				119	120	99	118	98	110	44	59	58	61	-116	-55	-65							
Shifted				87	88	67	86	66	78	63	46	47	43	-122	-69	-79							
Ctrl				23	24	3	22	2	14	44	59	58	61	-128	-83	-93							
Ctrl-Shifted				23	24	3	22	2	14	63	46	47	43	-134	-97	-107							
				SPACEBAR																			
				Unshifted												32							
				Shifted												-52							
				Ctrl												-53							
				Ctrl-Shifted												-54							

4893-21

Figure A-6. French Keyboard Macro Chart.

SWEDISH CHARACTER SET

BITS				0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
B7	B6	B5	B4	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
0	0	0	0								
0	0	0	0	NU ₀	DL ₁₆	Sp ₃₂	0 ₄₈	@ ₆₄	P ₈₀	\ ₉₆	p ₁₁₂
0	0	0	1	SH ₁	D1 ₁₇	! ₃₃	1 ₄₉	A ₆₅	Q ₈₁	a ₉₇	q ₁₁₃
0	0	1	0	SX ₂	D2 ₁₈	" ₃₄	2 ₅₀	B ₆₆	R ₈₂	b ₉₈	r ₁₁₄
0	0	1	1	EX ₃	D3 ₁₉	# ₃₅	3 ₅₁	C ₆₇	S ₈₃	c ₉₉	s ₁₁₅
0	1	0	0	ET ₄	D4 ₂₀	α ₃₆	4 ₅₂	D ₆₈	T ₈₄	d ₁₀₀	t ₁₁₆
0	1	0	1	EQ ₅	NK ₂₁	% ₃₇	5 ₅₃	E ₆₉	U ₈₅	e ₁₀₁	u ₁₁₇
0	1	1	0	AK ₆	SY ₂₂	& ₃₈	6 ₅₄	F ₇₀	V ₈₆	f ₁₀₂	v ₁₁₈
0	1	1	1	BL ₇	EB ₂₃	/ ₃₉	7 ₅₅	G ₇₁	W ₈₇	g ₁₀₃	w ₁₁₉
1	0	0	0	BS ₈	CN ₂₄	(₄₀	8 ₅₆	H ₇₂	X ₈₈	h ₁₀₄	x ₁₂₀
1	0	0	1	HT ₉	EM ₂₅) ₄₁	9 ₅₇	I ₇₃	Y ₈₉	i ₁₀₅	y ₁₂₁
1	0	1	0	LF ₁₀	SB ₂₆	* ₄₂	: ₅₈	J ₇₄	Z ₉₀	j ₁₀₆	z ₁₂₂
1	0	1	1	VT ₁₁	EC ₂₇	+ ₄₃	; ₅₉	K ₇₅	Ä ₉₁	k ₁₀₇	ä ₁₂₃
1	1	0	0	FF ₁₂	FS ₂₈	, ₄₄	< ₆₀	L ₇₆	Ö ₉₂	l ₁₀₈	ö ₁₂₄
1	1	0	1	CR ₁₃	GS ₂₉	- ₄₅	= ₆₁	M ₇₇	Å ₉₃	m ₁₀₉	å ₁₂₅
1	1	1	0	SO ₁₄	RS ₃₀	. ₄₆	> ₆₂	N ₇₈	^ ₉₄	n ₁₁₀	- ₁₂₆
1	1	1	1	SI ₁₅	US ₃₁	/ ₄₇	? ₆₃	O ₇₉	— ₉₅	o ₁₁₁	DT ₁₂₇

(4526)4893-24

Figure A-7. Swedish Character Set Code Chart.

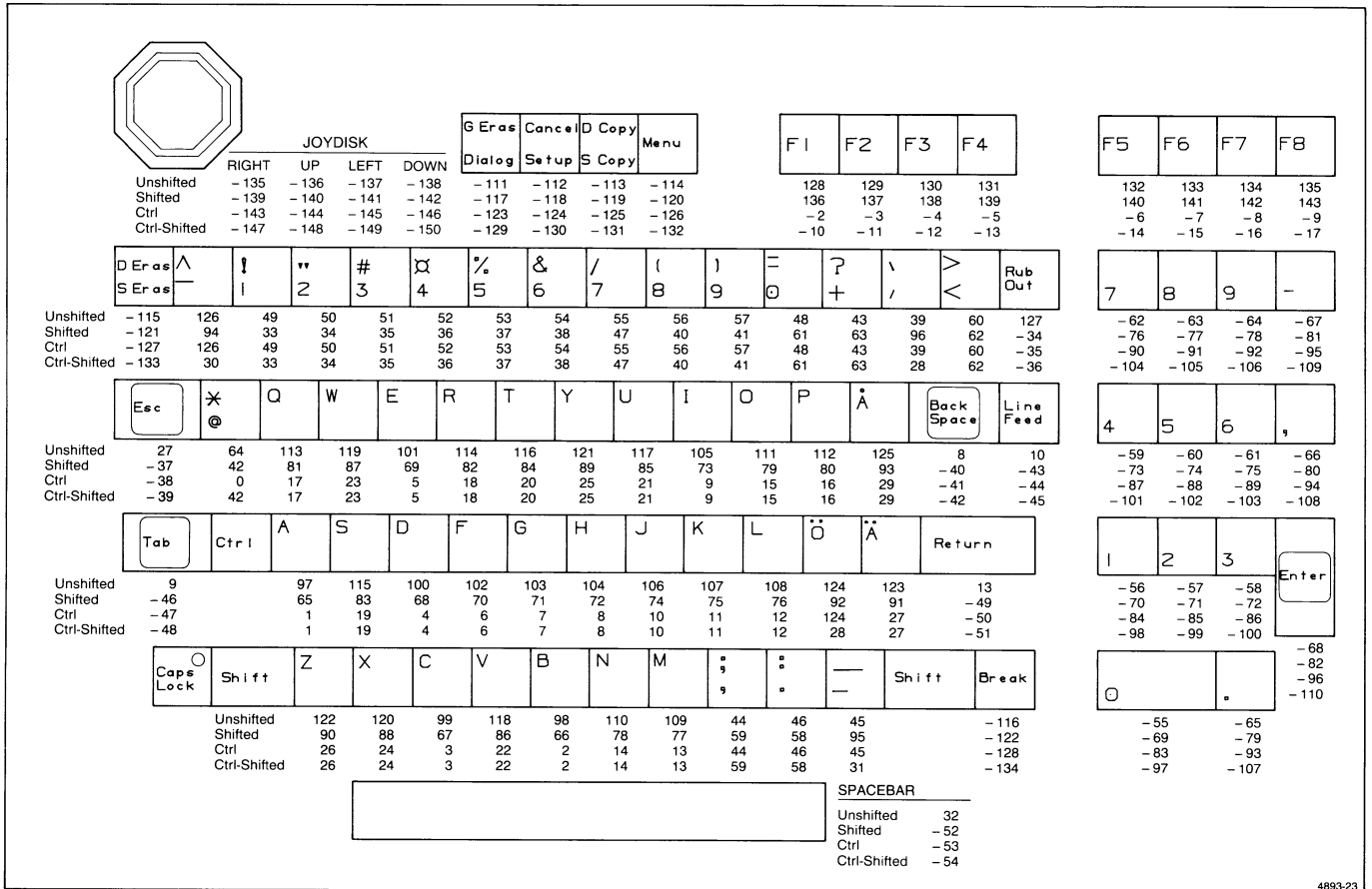


Figure A-8. Swedish Keyboard Macro Chart.

DANISH/NORWEGIAN CHARACTER SET

BITS				0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
B7	B6	B5	B4	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
0	0	0	0								
0	0	0	0	NU ₀	DL ₁₆	Sp ₃₂	0 ₄₈	@ ₆₄	P ₈₀	\ ₉₆	p ₁₁₂
0	0	0	1	SH ₁	D1 ₁₇	! ₃₃	1 ₄₉	A ₆₅	Q ₈₁	a ₉₇	q ₁₁₃
0	0	1	0	SX ₂	D2 ₁₈	" ₃₄	2 ₅₀	B ₆₆	R ₈₂	b ₉₈	r ₁₁₄
0	0	1	1	EX ₃	D3 ₁₉	# ₃₅	3 ₅₁	C ₆₇	S ₈₃	c ₉₉	s ₁₁₅
0	1	0	0	ET ₄	D4 ₂₀	\$ ₃₆	4 ₅₂	D ₆₈	T ₈₄	d ₁₀₀	t ₁₁₆
0	1	0	1	EQ ₅	NK ₂₁	% ₃₇	5 ₅₃	E ₆₉	U ₈₅	e ₁₀₁	u ₁₁₇
0	1	1	0	AK ₆	SY ₂₂	& ₃₈	6 ₅₄	F ₇₀	V ₈₆	f ₁₀₂	v ₁₁₈
0	1	1	1	BL ₇	EB ₂₃	/ ₃₉	7 ₅₅	G ₇₁	W ₈₇	g ₁₀₃	w ₁₁₉
1	0	0	0	BS ₈	CN ₂₄	(₄₀	8 ₅₆	H ₇₂	X ₈₈	h ₁₀₄	x ₁₂₀
1	0	0	1	HT ₉	EM ₂₅) ₄₁	9 ₅₇	I ₇₃	Y ₈₉	i ₁₀₅	y ₁₂₁
1	0	1	0	LF ₁₀	SB ₂₆	* ₄₂	: ₅₈	J ₇₄	Z ₉₀	j ₁₀₆	z ₁₂₂
1	0	1	1	VT ₁₁	EC ₂₇	+ ₄₃	; ₅₉	K ₇₅	Æ ₉₁	k ₁₀₇	æ ₁₂₃
1	1	0	0	FF ₁₂	FS ₂₈	, ₄₄	< ₆₀	L ₇₆	Ø ₉₂	l ₁₀₈	ø ₁₂₄
1	1	0	1	CR ₁₃	GS ₂₉	- ₄₅	= ₆₁	M ₇₇	Å ₉₃	m ₁₀₉	å ₁₂₅
1	1	1	0	SO ₁₄	RS ₃₀	. ₄₆	> ₆₂	N ₇₈	^ ₉₄	n ₁₁₀	- ₁₂₆
1	1	1	1	SI ₁₅	US ₃₁	/ ₄₇	? ₆₃	O ₇₉	_ ₉₅	o ₁₁₁	DT ₁₂₇

(4526)4893-26

Figure A-9. Danish/Norwegian Character Set Code Chart.

JOYDISK																F1 F2 F3 F4				F5 F6 F7 F8				
				RIGHT	UP	LEFT	DOWN	G Eras	Cancel	D Copy	Menu													
				Dialog	Setup	S Copy	Menu																	
Unshifted	-135	-136	-137	-138	-111	-112	-113	-114	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
Shifted	-139	-140	-141	-142	-117	-118	-119	-120	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12	-13	-14	-15	-16	-17
Ctrl	-143	-144	-145	-146	-123	-124	-125	-126																
Ctrl-Shifted	-147	-148	-149	-150	-129	-130	-131	-132																

D Eras ^ " # \$ % & / () = ? \ / > Rub Out																								
Unshifted	-115	126	49	50	51	52	53	54	55	56	57	48	43	39	60	127	7	8	9	-	-62	-63	-64	-67
Shifted	-121	94	33	34	35	36	37	38	47	40	41	61	63	96	62	-34	-76	-77	-78	-81	-76	-77	-78	-81
Ctrl	-127	126	49	50	51	52	53	54	55	56	57	48	43	39	60	-35	-90	-91	-92	-95	-90	-91	-92	-95
Ctrl-Shifted	-133	30	33	34	35	36	37	38	47	40	41	61	63	28	62	-36	-104	-105	-106	-109	-104	-105	-106	-109

Esc * Q W E R T Y U I O P A Back Space Line Feed																							
Unshifted	27	64	113	119	101	114	116	121	117	105	111	112	125	8	10	4	5	6	,	-59	-60	-61	-66
Shifted	-37	42	81	87	69	82	84	89	85	73	79	80	93	-40	-43	-73	-74	-75	-80	-73	-74	-75	-80
Ctrl	-38	0	17	23	5	18	20	25	21	9	15	16	29	-41	-44	-87	-88	-89	-94	-87	-88	-89	-94
Ctrl-Shifted	-39	42	17	23	5	18	20	25	21	9	15	16	29	-42	-45	-101	-102	-103	-108	-101	-102	-103	-108

Tab Ctrl A S D F G H J K L Ø Æ Return																					
Unshifted	9	97	115	100	102	103	104	106	107	108	124	123	13	1	2	3	Enter	-56	-57	-58	-68
Shifted	-46	65	83	68	70	71	72	74	75	76	92	91	-49	-70	-71	-72		-70	-71	-72	-82
Ctrl	-47	1	19	4	6	7	8	10	11	12	124	27	-50	-84	-85	-86		-84	-85	-86	-96
Ctrl-Shifted	-48	1	19	4	6	7	8	10	11	12	28	27	-51	-98	-99	-100		-98	-99	-100	-110

Caps Lock Shift Z X C V B N M ; , ' - _ Shift Break																	
Unshifted	122	120	99	118	98	110	109	44	46	45	-116					-55	-65
Shifted	90	88	67	86	66	78	77	59	58	95	-122					-69	-79
Ctrl	26	24	3	22	2	14	13	44	46	45	-128					-83	-93
Ctrl-Shifted	26	24	3	22	2	14	13	59	58	31	-134					-97	-107

SPACEBAR	
Unshifted	32
Shifted	-52
Ctrl	-53
Ctrl-Shifted	-54

4893-25

Figure A-10. Danish/Norwegian Keyboard Macro Chart.

GERMAN CHARACTER SET

BITS				0 0 0	0 0 1	0 1 0	0 1 1	1 0 0	1 0 1	1 1 0	1 1 1
B7	B6	B5		CONTROL		FIGURES		UPPERCASE		LOWERCASE	
0	0	0	0								
B4	B3	B2	B1								
0	0	0	0	NU ₀	DL ₁₆	Sp ₃₂	0 ₄₈	§ ₆₄	P ₈₀	\ ₉₆	p ₁₁₂
0	0	0	1	SH ₁	D1 ₁₇	! ₃₃	1 ₄₉	A ₆₅	Q ₈₁	a ₉₇	q ₁₁₃
0	0	1	0	SX ₂	D2 ₁₈	" ₃₄	2 ₅₀	B ₆₆	R ₈₂	b ₉₈	r ₁₁₄
0	0	1	1	EX ₃	D3 ₁₉	# ₃₅	3 ₅₁	C ₆₇	S ₈₃	c ₉₉	s ₁₁₅
0	1	0	0	ET ₄	D4 ₂₀	\$ ₃₆	4 ₅₂	D ₆₈	T ₈₄	d ₁₀₀	t ₁₁₆
0	1	0	1	EQ ₅	NK ₂₁	% ₃₇	5 ₅₃	E ₆₉	U ₈₅	e ₁₀₁	u ₁₁₇
0	1	1	0	AK ₆	SY ₂₂	& ₃₈	6 ₅₄	F ₇₀	V ₈₆	f ₁₀₂	v ₁₁₈
0	1	1	1	BL ₇	EB ₂₃	/ ₃₉	7 ₅₅	G ₇₁	W ₈₇	g ₁₀₃	w ₁₁₉
1	0	0	0	BS ₈	CN ₂₄	(₄₀	8 ₅₆	H ₇₂	X ₈₈	h ₁₀₄	x ₁₂₀
1	0	0	1	HT ₉	EM ₂₅) ₄₁	9 ₅₇	I ₇₃	Y ₈₉	i ₁₀₅	y ₁₂₁
1	0	1	0	LF ₁₀	SB ₂₆	* ₄₂	: ₅₈	J ₇₄	Z ₉₀	j ₁₀₆	z ₁₂₂
1	0	1	1	VT ₁₁	EC ₂₇	+ ₄₃	; ₅₉	K ₇₅	Ä ₉₁	k ₁₀₇	ä ₁₂₃
1	1	0	0	FF ₁₂	FS ₂₈	, ₄₄	< ₆₀	L ₇₆	Ö ₉₂	l ₁₀₈	ö ₁₂₄
1	1	0	1	CR ₁₃	GS ₂₉	- ₄₅	= ₆₁	M ₇₇	Ü ₉₃	m ₁₀₉	ü ₁₂₅
1	1	1	0	SO ₁₄	RS ₃₀	. ₄₆	> ₆₂	N ₇₈	^ ₉₄	n ₁₁₀	ß ₁₂₆
1	1	1	1	SI ₁₅	US ₃₁	/ ₄₇	? ₆₃	O ₇₉	— ₉₅	o ₁₁₁	DT ₁₂₇

(4526)4893-28A

Figure A-11. German Character Set Code Chart.
















SUPPLEMENTARY CHARACTER SET

BITS				$\emptyset \emptyset \emptyset$	$\emptyset \emptyset 1$	$\emptyset 1 \emptyset$	$\emptyset 1 1$	$1 \emptyset \emptyset$	$1 \emptyset 1$	$1 1 \emptyset$	$1 1 1$
B7	B6	B5	B4	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
\emptyset	\emptyset	\emptyset	\emptyset								
\emptyset	\emptyset	\emptyset	\emptyset	NU ₀	DL ₁₆	Sp ₃₂	0 ₄₈	— ₆₄	Ñ ₈₀	◆ ₉₆	▢ ₁₁₂
\emptyset	\emptyset	\emptyset	1	SH ₁	D1 ₁₇	Ä ₃₃	1 ₄₉	¢ ₆₅	ñ ₈₁	■ ₉₇	▣ ₁₁₃
\emptyset	\emptyset	1	\emptyset	SX ₂	D2 ₁₈	ä ₃₄	2 ₅₀	¡ ₆₆	¿ ₈₂	HT ₉₈	▤ ₁₁₄
\emptyset	\emptyset	1	1	EX ₃	D3 ₁₉	Å ₃₅	3 ₅₁	† ₆₇	ı ₈₃	FF ₉₉	▥ ₁₁₅
\emptyset	1	\emptyset	\emptyset	ET ₄	D4 ₂₀	å ₃₆	4 ₅₂	□ ₆₈	α ₈₄	CR ₁₀₀	▦ ₁₁₆
\emptyset	1	\emptyset	1	EQ ₅	NK ₂₁	Æ ₃₇	5 ₅₃	■ ₆₉	σ ₈₅	LF ₁₀₁	▧ ₁₁₇
\emptyset	1	1	\emptyset	AK ₆	SY ₂₂	æ ₃₈	6 ₅₄	● ₇₀	τ ₈₆	° ₁₀₂	▨ ₁₁₈
\emptyset	1	1	1	BL ₇	EB ₂₃	à ₃₉	7 ₅₅	△ ₇₁	ψ ₈₇	± ₁₀₃	▩ ₁₁₉
1	\emptyset	\emptyset	\emptyset	BS ₈	CN ₂₄	ç ₄₀	8 ₅₆	∂ ₇₂	μ ₈₈	NL ₁₀₄	▪ ₁₂₀
1	\emptyset	\emptyset	1	HT ₉	EM ₂₅	é ₄₁	9 ₅₇	λ ₇₃	Σ ₈₉	VT ₁₀₅	≤ ₁₂₁
1	\emptyset	1	\emptyset	LF ₁₀	SB ₂₆	è ₄₂	ù ₅₈	⌋ ₇₄	Ω ₉₀	▫ ₁₀₆	≥ ₁₂₂
1	\emptyset	1	1	VT ₁₁	EC ₂₇	ö ₄₃	β ₅₉	⌌ ₇₅	∫ ₉₁	▬ ₁₀₇	π ₁₂₃
1	1	\emptyset	\emptyset	FF ₁₂	FS ₂₈	ö ₄₄	θ ₆₀	⌍ ₇₆	∫ ₉₂	▮ ₁₀₈	≠ ₁₂₄
1	1	\emptyset	1	CR ₁₃	GS ₂₉	ø ₄₅	ϝ ₆₁	⌎ ₇₇	÷ ₉₃	▯ ₁₀₉	£ ₁₂₅
1	1	1	\emptyset	SO ₁₄	RS ₃₀	ü ₄₆	§ ₆₂	⌏ ₇₈	≈ ₉₄	▰ ₁₁₀	• ₁₂₆
1	1	1	1	SI ₁₅	US ₃₁	ü ₄₇	•• ₆₃	∞ ₇₉	√ ₉₅	▱ ₁₁₁	DT ₁₂₇

(4526)4893-29C

Figure A-13. Supplementary Character Set Code Chart.

RULINGS CHARACTER SET

BITS				$\emptyset \emptyset \emptyset$	$\emptyset \emptyset 1$	$\emptyset 1 \emptyset$	$\emptyset 1 1$	$1 \emptyset \emptyset$	$1 \emptyset 1$	$1 1 \emptyset$	$1 1 1$			
B7	B6	B5	B4	B3	B2	B1	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
\emptyset	\emptyset	\emptyset	\emptyset				NU ₀	DL ₁₆	Sp ₃₂	0 ₄₈	@ ₆₄	P ₈₀	◆ ₉₆	 ₁₁₂
\emptyset	\emptyset	\emptyset	1				SH ₁	D1 ₁₇	! ₃₃	1 ₄₉	A ₆₅	Q ₈₁	■ ₉₇	 ₁₁₃
\emptyset	\emptyset	1	\emptyset				SX ₂	D2 ₁₈	" ₃₄	2 ₅₀	B ₆₆	R ₈₂	HT ₉₈	 ₁₁₄
\emptyset	\emptyset	1	1				EX ₃	D3 ₁₉	# ₃₅	3 ₅₁	C ₆₇	S ₈₃	FF ₉₉	 ₁₁₅
\emptyset	1	\emptyset	\emptyset				ET ₄	D4 ₂₀	\$ ₃₆	4 ₅₂	D ₆₈	T ₈₄	CR ₁₀₀	 ₁₁₆
\emptyset	1	\emptyset	1				EQ ₅	NK ₂₁	% ₃₇	5 ₅₃	E ₆₉	U ₈₅	LF ₁₀₁	 ₁₁₇
\emptyset	1	1	\emptyset				AK ₆	SY ₂₂	& ₃₈	6 ₅₄	F ₇₀	V ₈₆	° ₁₀₂	 ₁₁₈
\emptyset	1	1	1				BL ₇	EB ₂₃	/ ₃₉	7 ₅₅	G ₇₁	W ₈₇	± ₁₀₃	 ₁₁₉
1	\emptyset	\emptyset	\emptyset				BS ₈	CN ₂₄	(₄₀	8 ₅₆	H ₇₂	X ₈₈	NL ₁₀₄	 ₁₂₀
1	\emptyset	\emptyset	1				HT ₉	EM ₂₅) ₄₁	9 ₅₇	I ₇₃	Y ₈₉	VT ₁₀₅	≤ ₁₂₁
1	\emptyset	1	\emptyset				LF ₁₀	SB ₂₆	* ₄₂	: ₅₈	J ₇₄	Z ₉₀	 ₁₀₆	≥ ₁₂₂
1	\emptyset	1	1				VT ₁₁	EC ₂₇	+ ₄₃	; ₅₉	K ₇₅	[₉₁	 ₁₀₇	π ₁₂₃
1	1	\emptyset	\emptyset				FF ₁₂	FS ₂₈	, ₄₄	< ₆₀	L ₇₆	\ ₉₂	 ₁₀₈	≠ ₁₂₄
1	1	\emptyset	1				CR ₁₃	GS ₂₉	- ₄₅	= ₆₁	M ₇₇] ₉₃	 ₁₀₉	£ ₁₂₅
1	1	1	\emptyset				SO ₁₄	RS ₃₀	. ₄₆	> ₆₂	N ₇₈	^ ₉₄	 ₁₁₀	• ₁₂₆
1	1	1	1				SI ₁₅	US ₃₁	/ ₄₇	? ₆₃	O ₇₉		 ₁₁₁	DT ₁₂₇

(4526)4893-30

Figure A-14. Rulings Character Set Code Chart.

Appendix B

ERROR CODES

INTRODUCTION

Each error that the terminal detects has an *error code* and a *severity level*.

When the terminal detects an error, it stores the error in a limited-size queue for later retrieval by the REPORT ERRORS command.

If the error's severity level is greater than or equal to the current error level, the terminal displays a message on the screen. When the terminal is shipped from the factory, its error threshold is set to 2; thus the only errors displayed are those with a severity level of 2 or 3. The error threshold can be changed with the SET ERROR THRESHOLD command (ERRORLEVEL in Setup syntax). The error threshold is not remembered when the terminal is turned off.

ERROR CODES

The error codes are composed according to the following scheme:

- Each error code consists of four characters.
- In most error codes the first two characters are the opcode, which identifies the command that caused the error.
- The third character is a digit. Digits from 1 to 9 name the parameter with which the error is associated; a 0 indicates that the error is associated with the command as a whole.
- The fourth character of the error code is also a digit:
 - 0 — Indicates an *existence problem*. The object referred to does not exist when it should, or it does exist when it shouldn't.
 - 1 — Indicates an *invalid value*.
 - 2 — Indicates an *out-of-memory problem*.
 - 3 — Indicates a *context problem*. The command is valid, but cannot be executed at this time.

For example, consider the MP10 error code. Here *MP* refers to the SELECT FILL PATTERN command, which has the following syntax:

```
ⒺcMP fill-pattern-number
```

The *I* refers to the first (and only) parameter of that command, which is the fill pattern number. The *0* indicates an existence problem; the fill pattern does not exist.

SEVERITY LEVELS

There are four error severity levels:

- **Level 0.** Level errors 0 are minor errors. The corresponding message begins with *Terminal issues message*, followed by the error code.
- **Level 1.** Level 1 errors are warnings. The corresponding message begins with *Terminal issues warning*, followed by the error code. Typically these warnings occur when the command is inappropriate or not recognized.
- **Level 2.** Level 2 errors result from invalid commands. The corresponding message begins with *Terminal detects error*, followed by the error code. For instance, a parameter may be outside the specified range.
- **Level 3.** Level 3 errors occur when the command is valid, but the terminal cannot execute the command. The corresponding message begins with *Terminal system error* followed by the error code. For instance, there may be insufficient memory to hold all the information being entered.

ERROR CODES

ERROR CODES

The rest of this appendix lists each error code (in bold) alphabetically with its severity level and an explanation of the cause.

ERROR CODES FOR 4100-STYLE COMMANDS

IJ — SET GIN CURSOR SPEED

IJ11 (Level 2). Invalid *normal-speed* parameter (must be in the range 0 through 65535).

IJ21 (Level 2). Invalid *shifted-speed* parameter (must be in the range 0 through 65535).

JC — COPY

JC11 (Level 2). Invalid *source* parameter (must be HO:).

JC12 (Level 3). Out of memory while parsing the *source* parameter.

JC21 (Level 2). Invalid *separator* parameter (must be empty string or TO).

JC22 (Level 3). Out of memory while parsing the *separator* parameter.

JC31 (Level 2). Either the port is busy or the *destination* parameter is invalid (must be HC:).

JC32 (Level 3). Out of memory while parsing the *destination* parameter.

JC39 (Level 2). Hard copy device not ready. Check the hard copy unit.

KA — ENABLE DIALOG AREA

KA03 (Level 1). Context problem, cannot disable dialog area when in ANSI mode.

KA11 (Level 2). Invalid *enable-mode* parameter (must be 0 or 1).

KB — SET TAB STOPS

KB12 (Level 2). Out of memory while parsing the array.

KD — DEFINE MACRO

KD11 (Level 2). Invalid *macro-number* parameter (must be in range -150 to 32767).

KD21 (Level 2). Invalid ADE integer in the *macro-contents* parameter. (Character codes must be in the range from 0 to 127. The array count must be in the range from 0 to 65535.)

KD22 (Level 3). Insufficient memory to define macro.

KE — SET ECHO

KE11 (Level 2). Invalid *echo-mode* parameter (must be 0 or 1).

KF — LFCR

KF11 (Level 2). Invalid *LFCR-mode* parameter (must be 0 or 1).

KG — DIM ENABLE

KG11 (Level 2). Invalid *dim-code* parameter (must be 0 or 1).

KH — HARDCOPY

KH11 (Level 2). Invalid *hardcopy-code* parameter (must be 0, 1, 2, or 3). (Or, *hardcopy-code* does not agree with current setting of SELECT HARDCOPY INTERFACE command.)

KI — IGNORE DELETES

KI11 (Level 2). Invalid *ignore-deletes-mode* parameter (must be 0 or 1).

KL — LOCK KEYBOARD

KL11 (Level 2). Invalid *locking-mode* parameter (must be 0 or 1).

KO — DEFINE NONVOLATILE MACRO

KO11 (Level 2). Invalid *macro-number* parameter (must be in range -150 through 32767).

KO21 (Level 2). Invalid ADE integer in the *macro-contents* parameter (character codes must be in the range 0 through 127; the array count must be in the range 0 through 65535).

KO22 (Level 3). Insufficient memory to define macro.

KR — CRLF

KR11 (Level 2). Invalid *CRLF-mode* parameter (must be 0 or 1).

KS — SET SNOOPY MODE

KS11 (Level 2). Invalid *snoopy-mode* parameter (must be 0 or 1).

KT — SET ERROR THRESHOLD

KT11 (Level 2). Invalid *error-threshold-level* parameter (must be in range from 0 to 4).

KU — SAVE NONVOLATILE PARAMETERS

KU02 (Level 2). Nonvolatile memory hardware error (data was not saved correctly in nonvolatile memory; defaults will be reset on power-up).

KW — ENABLE KEY EXPANSION

KW11 (Level 2). Invalid *macro-expansion-mode* parameter (must be 0 or 1).

KX — EXPAND MACRO

KX01 (Level 2). The maximum nesting depth for the command has been exceeded (the nesting depth should not exceed five).

KX02 (Level 3). Out of memory while executing command.

KX11 (Level 2). Invalid *macro-number* parameter (must be in the range -150 to 32767).

KY — SET KEY EXECUTE CHARACTER

KY11 (Level 2). Invalid *key-execute-character* parameter (must be in range 0 to 127).

KZ — SET EDIT CHARACTERS

KZ11 (Level 2). Invalid *character-delete* parameter (must range from 0 to 127).

KZ21 (Level 2). Invalid *line-delete* parameter (must range from 0 to 127).

KZ31 (Level 2). Invalid *take-literally* parameter (must range from 0 to 127).

LB — SET DIALOG AREA BUFFER SIZE

LB03 (Level 0). Context error: dialog parameters modified, but not as specified.

LB11 (Level 2). Invalid *number-of-lines* parameter (must be in the range from 2 to 32767).

LE — END PANEL

LE02 (Level 3). Out of memory while executing command.

LE03 (Level 1). No panel is currently being defined.

LI — SET DIALOG AREA INDEX

LI11 (Level 2). Invalid *character-index* parameter (must be in the range from 0 to 65535).

LI21 (Level 2). Invalid *character-background-index* parameter (must be in the range from 0 to 65535).

LI31 (Level 2). Invalid *dialog-background-index* parameter (must be in the range from 0 to 65535).

LL — SET DIALOG AREA LINES

LL11 (Level 2). Invalid *number-of-lines* parameter (must be in the range of 2 through 30).

LM — SET DIALOG AREA WRITING MODE

LM11 (Level 2). Invalid *writing-mode* parameter (must be 0 or 1).

ERROR CODES

LP — BEGIN PANEL BOUNDARY

LP03 (Level 2). Alphatext is not allowed within a panel definition. If the dialog area is disabled when the terminal is in Alpha mode, the terminal attempts to read alphatext as xy parameters.

LP21 (Level 2). Invalid *draw-boundary* parameter (must be 0 or 1).

LT — GRAPHIC TEXT

LT03 (Level 2). Command is invalid at this time. Graphtext is not allowed within a panel definition.

LT11 (Level 2). Invalid *text* parameter. Invalid array count (must be in range from 0 to 32767), or invalid character in the array (must be in the range ADE 32 through 126, ^{SP} through ~).

LT12 (Level 3). Out of memory while parsing the *text* parameter).

LV — SET DIALOG AREA VISIBILITY

LV11 (Level 2). Invalid *visibility-mode* parameter (must be 0 or 1).

MC — SET GRAPHTEXT SIZE

MC11 (Level 2). Invalid *width* parameter (must be in range 0 through 4095).

MC21 (Level 2). Invalid *height* parameter (must be in range 0 through 4095).

MC31 (Level 2). Invalid *spacing* parameter (must be in range 0 through 4095).

MG — SET GRAPHICS AREA WRITING MODE

MG11 (Level 2). Invalid *writing-mode* parameter (must be 0 or 1).

ML — SET LINE INDEX

ML11 (Level 2). Invalid *line-index* parameter (must be in the range from 0 to 65535).

MM — SET MARKER TYPE

MM11 (Level 2). Invalid *marker-number* parameter (must be in the range from 0 to 10).

MN — SET GRAPHTEXT CHARACTER PATH

MN11 (Level 2). Invalid *direction* parameter (must be 0, 1, 2, or 3).

MP — SELECT FILL PATTERN

MP10 (Level 2). Specified fill pattern does not exist.

MP11 (Level 2). Invalid *fill-pattern-number* parameter (must be in the range from -32768 through 16, or 50 through 174).

MR — SET GRAPHTEXT ROTATION

MR11 (Level 2). Invalid angle (*mantissa* and *power-of-two* parameters must represent an angle in the range -32767.0 to 32767.0).

MT — SET TEXT INDEX

MT11 (Level 2). Invalid *text-index* parameter (must be in the range from 0 to 65535).

MV — SET LINE STYLE

MV11 (Level 2). Invalid *line-style* parameter (must be in the range from 0 to 7).

NB — SET STOP BITS

NB11 (Level 2). Invalid *number-of-stopbits* parameter (must be 1 or 2).

NC — SET EOM CHARACTERS

NC11 (Level 2). Invalid *first-EOM-character* parameter (must be an integer in the range 0 through 127).

NC21 (Level 2). Invalid *second-EOM-character* parameter (must be an integer in the range 0 through 127).

ND — SET TRANSMIT DELAY

ND11 (Level 2). Invalid *transmit-delay* parameter (must be in the range from 0 to 65535 milliseconds).

NE — SET EOF STRING

NE11 (Level 2). Invalid *EOF-string* parameter (must contain from 0 to 10 characters, and each integer in the array must be in the range from 0 to 127.)

NE12 (Level 3). Out of memory while parsing the *EOF-string* parameter.

NF — SET FLAGGING MODE

NF11 (Level 2). Invalid *flagging-mode* parameter (must be in the range from 0 to 4).

NK — SET BREAK TIME

NK11 (Level 2). Invalid *break-time* parameter (must be in the range from 0 to 65535).

NL — SET TRANSMIT RATE LIMIT

NL11 (Level 2). Invalid *rate-limit* parameter (must be in the range from 110 to 65535).

NM — PROMPT MODE

NM11 (Level 2). Invalid *prompt-mode* parameter (must be 0, 1, or 2).

NP — SET PARITY

NP11 (Level 2). Invalid *parity-code* parameter (must be in the range from 0 to 4).

NQ — SET QUEUE SIZE

NQ02 (Level 3). Out of memory while performing command.

NQ11 (Level 2). Invalid *queue-size* parameter (must be in the range from 1 to 65535).

NR — SET BAUD RATES

NR11 (Level 2). Invalid *transmit-data-rate* (must be either 1, 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600, 19200, or 38,400).

NR21 (Level 2). Invalid *receive-data-rate* (must be either 0, 1, 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600, or 19200, or 38,400).

NS — SET PROMPT STRING

NS11 (Level 2). Invalid *prompt-string* parameter (must be an array of up to ten integers, each in the range 0 through 127).

NS12 (Level 3). Out of memory while parsing the *prompt-string* parameter.

NT — SET EOL STRING

NT11 (Level 2). Invalid *EOL-string* parameter. The array must hold from 0 to 2 integer parameters. Each integer in the array must be in the range from 0 to 127.

NT12 (Level 3). Out of memory while parsing the *EOL-string* parameter).

NU — SET BYPASS CANCEL CHARACTER

NU11 (Level 2). Invalid ADE value in the *bypass-cancel-character* parameter (must be in the range from 0 to 127).

QA — SET COPY SIZE

QA11 (Level 2). Invalid *copy-size* parameter (must be 0 or 1).

QD — SELECT HARDCOPY INTERFACE

QD11 (Level 2). Invalid *copier-type* parameter (must be 0, 1, or 2).

QE — SET HARDCOPY MONOCHROME ATTRIBUTES

QE11 (Level 2): Invalid *monochrome-attributes-array* parameter must be (0 or 1).

QE12 (Level 3): Out of memory while parsing *monochrome-attributes-array*.

ERROR CODES

QI — MAP INDEX TO PRINT

QI11 (Level 2). Invalid *monochrome-values* array.

QI12 (Level 3). Out of memory while parsing *monochrome-values* array.

QL — SET DIALOG AREA HARDCOPY ATTRIBUTES

QL11 (Level 2). Invalid *number-of-pages* parameter (must be in the range 0 through 32767).

QL21 (Level 2). Invalid *page-origin* parameter (must be 0, 1, or 2).

QL31 (Level 2). Invalid *FF-interpretation* parameter (must be 0, 1, or 2).

QO — SET IMAGE ORIENTATION

QO11 (Level 2). Invalid *orientation* parameter (must be in the range 0 through 3).

QT — SET COLOR COPIER REPAINT

QT11 (Level 2). Invalid *repaint-count* parameter (must be between 0 and 4).

QU — SELECT COLOR HARDCOPY IMAGE DENSITY

QU11 (Level 2). Invalid *density-code* parameter (must be 0 or 1).

RA — SET VIEW ATTRIBUTES

RA11 (Level 2). Invalid *surface-number* parameter (must be -1 or 1).

RA21 (Level 2). Invalid *wipe-index* parameter (must be in range 0 through 65535).

RA31 (Level 2). Invalid *border-index* parameter (must be in range 0 through 65535).

RL — RUNLENGTH WRITE

RL11 (Level 2). Invalid *runcode-array* (the array count must range from 0 to 65535, and each integer in the array must also range from 0 to 65535).

RL12 (Level 3). Out of memory while parsing the *runcode-array* parameter, or while executing the command.

RP — RASTER WRITE

RP11 (Level 2). Invalid *number-of-pixels* parameter (must be in the range 0 through 65535).

RP21 (Level 2). There are too many or too few pixels in the *color-index-codes* array.

RP22 (Level 3). Out of memory while parsing the *color-index-codes* array parameter.

RR — RECTANGLE FILL

RR11 (Levels 0 and 2):

(Level 0). Invalid *lower-left-corner* coordinates (x value is between range of 480 and 511, which is off-screen memory).

(Level 2). Invalid *lower-left-corner* coordinates (x must be in the range 0 through 511, and y from 0 through 359).

RR21 (Levels 0 and 2):

(Level 0). Invalid *upper-right-corner* coordinates (x value is between range of 480 and 511, which is off-screen memory).

(Level 2). Invalid *upper-right-corner* coordinates (x must be in the range 0 through 511, and y from 0 through 359).

RR31 (Level 2). Invalid *fill-index* parameter (must be in range 0 through 65535).

RS — SET PIXEL VIEWPORT

RS11 (Levels 0 and 2):

(Level 0). Invalid *lower-left* coordinate (x value is between range of 480 and 511, which is off-screen memory).

(Level 2). Invalid *lower-left* coordinate (x must be in the range 0 through 479, and y from 0 through 359).

RS21 (Levels 0 and 2):

(Level 0). Invalid *upper-right* coordinate (x value is between range of 480 and 511, which is off-screen memory).

(Level 2). Invalid *upper-right* coordinate (x must be in the range 0 through 479, and y from 0 through 359).

RU — BEGIN PIXEL OPERATIONS

RU11 (Level 2). Invalid *surface-number* parameter (must be in the range -1 through 1).

RU21 (Level 2). Invalid *ALU-mode* parameter (must be 0, 7, 11, 12, 15).

RU31 (Level 2). Invalid *bits-per-pixel* parameter (must be 0, 1, 2, 3, 4, or 6).

RX — PIXEL COPY

RX11 (Level 2). Invalid *destination-surface* parameter (must be in the range -1 through 1).

RX21 (Levels 0 and 2):

(Level 0). Invalid *destination-lower-left-corner* parameter (x must be in range 0 through 479 and y in range 0 through 359).

(Level 2). Invalid *destination-lower-left-corner* parameter (x must be in range 0 through 511 and y in range 0 through 359).

RX31 (Levels 0 and 2):

(Level 0). Invalid *first-source-corner* parameter (x must be in range 0 through 479 and y in range 0 through 359).

(Level 2). Invalid *first-source-corner* parameter (x must be in range 0 through 511 and y in range 0 through 359).

RX41 (Levels 0 and 2):

(Level 0). Invalid *second-source-corner* parameter (x must be in range 0 through 479 and y in range 0 through 359).

(Level 2). Invalid *second-source-corner* parameter (x must be in range 0 through 511 and y in range 0 through 359).

SX — SET SEGMENT POSITION

SX11 (Level 2). Invalid *segment-number* parameter (must be 0).

TC — SET GIN CURSOR COLOR

TC11 (Level 2). Invalid *first-color-coordinate* parameter (must be between -32768 and 32767).

TC21 (Level 2). Invalid *second-color-coordinate* parameter (must be between 0 and 100).

TC31 (Level 2). Invalid *third-color-coordinate* parameter (must be between 0 and 100).

TD — SET ALPHA CURSOR INDICES

TD11 (Level 2). Invalid *first-index* parameter (must be between 0 and 65535).

TD21 (Level 2). Invalid *second-index* parameter (must be between 0 and 65535).

TF — SET DIALOG AREA COLOR MAP

TF11 (Level 2). Invalid *color-mixtures* array (must be an array of quadruples, each consisting of a color index and three color coordinates).

TF12 (Level 3). Out of memory while parsing the *color-mixtures* array.

TG — SET SURFACE COLOR MAP

TG11 (Level 2). Invalid *surface-number* parameter (must be 1).

TG21 (Level 2). Invalid *color-mixtures* array (must be an array of quadruples consisting of a color index and the three HLS color coordinates).

TG22 (Level 3). Out of memory while parsing the *color-mixtures* array.

ERROR CODES

ERROR CODES FOR ANSI COMMANDS

[A — CUU (CURSOR UP)

[A11 (Level 2). Invalid *number-of-lines* parameter (must be in the range 0 to 32767).

[B — CUD (CURSOR DOWN)

[B11 (Level 2). Invalid *number-of-lines* parameter (must be in the range 0 to 32767).

[C — CUF (CURSOR FORWARD)

[C11 (Level 2). Invalid *number-of-columns* parameter (must be in the range 0 to 32767).

[c — DA (DEVICE ATTRIBUTES)

[c11 (Level 2). Invalid first parameter (must be 0 or omitted).

[D — CUB (CURSOR BACKWARD)

[D11 (Level 2). Invalid *number-of-columns* parameter (must be in the range 0 to 32767).

[f — HVP (HORIZONTAL & VERTICAL POSITION)

[f11 (Level 2). Invalid *row-number* parameter (must be in the range 0 to 32767).

[f21 (Level 2). Invalid *column-number* parameter (must be in the range 0 to 32767).

[g — TBC (TAB CLEAR)

[g11 (Level 2). Invalid *tab-clear-extent* parameter (must be 0, 2 or 3).

[H — CUP (CURSOR POSITION)

[H11 (Level 2). Invalid *row-number* parameter (must be in the range 0 to 32767).

[H21 (Level 2). Invalid *column-number* parameter (must be in the range 0 to 32767).

[h — SM (SET MODE)

[h01 (Level 2). Invalid *mode* parameter (must be 4, 20, < 1, ?1, ?3, ?6, ?7, or ?8).

[h10 to [h90 (Level 2). Invalid *mode* parameter.

[h11 to [h91 (Level 2). Invalid *mode* parameter syntax.

[h13 to [h93 (Level 0). Command is invalid at this time.

[I — CHT (CURSOR HORIZONTAL TAB)

[I11 (Level 2). Invalid *number-of-following-tab-stops* parameter (must be in the range 0 to 32767).

[i — (MEDIA COPY)

[i03 (Level 2). Printer not available.

[i11 to [i9 (Level 2). Invalid *copy-option* parameter value (the parameter must be 0, ?3, ?4, or ?5).

[J — ED (ERASE IN DISPLAY)

[J11 (Level 2). Invalid *erase-extent* parameter (must be in the range 0 through 32767).

[K — EL (ERASE IN LINE)

[K11 (Level 2). Invalid *erase-extent* parameter (must be in the range 0 through 32767).

[L — IL (INSERT LINE)

[L11 (Level 2). Invalid *number-of-lines* parameter (must be in the range 0 to 32767).

[I — RM (RESET MODE)

[I01 (Level 2). Invalid *mode* parameter (must be 4, 20, < 1, ?1, ?3, ?6, ?7, or ?8).

[I10 to [I90 (Level 0). Unrecognized *mode* parameter (treated as a no-op).

[I11 to [I91 (Level 0). Invalid *mode* parameter syntax.

[M — DL (DELETE LINE)

[M11 (Level 2). Invalid *number-of-lines* parameter (must be in the range 0 to 32767).

[m — SGR (SELECT GRAPHIC RENDITION)

[m01 (Level 2). Invalid *graphic-rendition parameter* parameter (must be 0, 1, 4, 5, 7, 24, 25, 27, 30 through 37, 39, 40 through 47, 49, < with parameter 0 through 255, > with parameter 0 through 255, = with parameter 0 through 255).

[n — DSR (DEVICE STATUS REPORT)

[n11 (Level 2). Invalid *status* parameter (must be 0, 5, or 6).

[P — DCH (DELETE CHARACTER)

[P11 (Level 2). Invalid *number-of-characters* parameter (must be in the range 0 to 32767).

[r — TEKSTBM (SET TOP & BOTTOM MARGINS)

[r11 (Level 2). Invalid *top-margin* parameter (must be in the range 0 to 32767).

[r21 (Level 2). Invalid *bottom-margin* parameter (must be in the range 0 to 32767).

[S — SU (SCROLL UP)

[S11 (Level 2). Invalid *number-of-lines* parameter (must be in the range 0 to 32767).

[T — SD (SCROLL DOWN)

[T11 (Level 2). Invalid *number-of-lines* parameter (must be in the range 0 to 32767).

[X — ECH (ERASE CHARACTER)

[X11 (Level 2). Invalid *number-of-characters* parameter (must be in the range 0 to 32767).

[Z — CBT (CURSOR BACKWARD TAB)

[Z11 (Level 2). Invalid *number-of-preceding-tab-stops* parameter (must be in the range 0 to 32767).

[^sP_A — SL (SCROLL LEFT)

^sP_A11 (Level 2). Invalid *number-of-columns* parameter (must be 0 to 32767).

[^sP@ — SR (SCROLL RIGHT)

^sP@11 (Level 2). Invalid *number-of-columns* parameter (must be 0 to 32767).

[@ — ICH (INSERT CHARACTER)

[@11 (Level 2). Invalid *number-of-characters* parameter (must be in the range 0 to 32767).

%! — SELECT CODE

%!11 (Level 2). Invalid *syntax* parameter (must be 0, 1, 2, or 3).

#!0 — REPORT SYNTAX MODE

#!11 (Level 2). Invalid parameter (must be 0).

Appendix C

COMMAND SUMMARY TABLES

This appendix provides a convenient summary of the commands available for your terminal. There are three command sets and three tables:

- Table C-1, *4100-Style Commands*
- Table C-2, *ANSI-Style Commands*
- Table C-3, *VT52-Style Commands*

Each table lists the host and Setup command syntax, parameter names, default parameter values, and function of each command.

All three tables use the following headings:

Descriptive Name — The name of the command as you will find it in the concepts discussions and command descriptions in this manual.

Setup Syntax — The name of the command as you enter it from the keyboard while in Setup.

Host Syntax — The escape sequence (or control character) that a host application uses to issue the command.

Parameters — The names of parameters used with each command. The valid values and keywords for each parameter are given in the command descriptions in Sections 3 and 5.

Defaults — The value the terminal assigns to a parameter when you do not enter it explicitly. A blank entry under the *Default* column means that the terminal does not assign a default value for that parameter. There are two possible defaults:

- **Factory** — The value assigned to a parameter as delivered from the factory. (The terminal reverts to this factory parameter value when you enter the **FACTORY** command.)
- **Omitted** — The value the terminal assigns to a parameter when you enter a Setup command but do not specify a value for the parameter. For example, entering *LOCAL* has the same effect as entering *LOCAL YES* (because *YES* is the default).

You cannot omit parameters when issuing a command from a host application.

Saved — Specifies whether or not the effect of the command can be saved with a **SAVE NONVOLATILE PARAMETER** command. The terminal remembers saved values of parameters at power-up. Parameters for commands that cannot be saved either revert to the factory default or have no power-up value.

Description — Provides a brief description of the command's function.

COMMAND SUMMARY

4100-STYLE COMMANDS

Table C-1 lists all the 4100-style commands, which are available to the host computer in TEK mode.

Table C-1
4100-STYLE COMMANDS

Descriptive Name	Setup Name	Host Syntax ^a	Parameters ^b	Defaults ^a		Saved	Description
				Factory	If Omitted ^b		
BEGIN PANEL BOUNDARY	BEGINPANEL	E _c LP	first-point		0,0	No	Starts the definition of a panel boundary
			draw-boundary		0		
BEGIN PIXEL OPERATIONS	PXBEGIN	E _c RU	surface-number	1	0	No	Sets up the terminal for pixel operations
			ALU-mode	11	0		
			bits-per-pixel	6	0		
CANCEL	CANCEL	E _c KC				No	Stops terminal activity and resets several terminal settings to their default values
CLEAR DIALOG SCROLL	CLEARDIALOG	E _c LZ				No	Erases the dialog area buffer
COPY	COPY HO: TO	E _c JC	destination		Error JC31	No	Sends data from the host directly to a copier or printer
CRLF	CRLF	E _c KR	crlf-mode	0	1	Yes	Specifies whether a C _R character sent to the terminal also implies an L _F
DEFINE MACRO	DEFINE	E _c KD	macro-number		0	No	Creates or deletes a volatile macro
			macro-contents		Empty array		
			string		Empty string		
DEFINE NONVOLATILE MACRO	NVDEFINE	E _c KO	macro-number		0	Yes	Creates or deletes nonvolatile macro, which can be saved in nonvolatile memory
			macro-contents		Empty array		
			string		Empty string		
DRAW	DRAW	E _c LG	position		0,0	No	Draws a vector from the current graphics position to a new position
DRAW MARKER	MARKER	E _c LH	position		0,0	No	Draws a marker at a specified location
ENABLE DIALOG AREA	DAENABLE	E _c KA	mode	1	1	Yes	Enables or disables the dialog area
ENABLE KEY EXPANSION	KEYEXPAND	E _c KW	mode	1	1	No	Enables or disables key macros
ENABLE 4010 GIN	(none)	E _c S _B				No	Enables the terminal for one 4010 GIN report
END PANEL	ENDPANEL	E _c LE				No	Concludes a panel definition
ENQUIRY	(none)	E _c Q				No	Queries the terminal for its answerback string
ENTER ALPHA MODE	(none)	U _S				No	Puts the terminal in Alpha mode

^a A blank entry indicates that there is no default value.

^b You can only omit parameters in Setup.

(continued)

Table C-1 (cont)
4100-STYLE COMMANDS

Descriptive Name	Setup Name	Host Syntax ^a	Parameters ^b	Defaults ^a		Saved	Description
				Factory	If Omitted ^b		
ENTER BYPASS MODE	(none)	E _c C _N				No	Puts the terminal in Bypass mode
ENTER MARKER MODE	(none)	F _S				No	Puts the terminal in Marker mode
ENTER VECTOR MODE	(none)	G _S				No	Puts the terminal in Vector mode
EXPAND MACRO	EXPAND	E _c K _X	macro-number		0	No	Expands a macro
FACTORY	FACTORY	(none)				No	Sets all parameters to their factory default values and takes the terminal out of Setup
GRAPHIC TEXT	GTEXT	E _c L _T	text		Empty array	No	Writes a string of graphtext, starting at the current graphics position
HARDCOPY	(none)	E _c K _H	hardcopy-code		0	No	Copies the terminal's screen (or just the dialog area) to a copier or printer
HELP	HELP	(none)	name		all commands	No	Displays information about a command or cluster of commands
IGNORE DELETES	IGNOREDEL	E _c K _I	ignore-deletes-mode	0	1	Yes	Determines whether the terminal ignores the ^D _T (Delete) character
LEARN	LEARN	(none)				No	Programs a key from the keyboard
LEARN NONVOLATILE	NVLEARN	(none)				Yes	Programs a key from the keyboard so that the definition can be saved in nonvolatile memory
LFCR	LFCR	E _c K _F	lfcf-mode	0	1	Yes	Specifies whether an ^L _F character sent to the terminal also implies a ^C _R
LOCAL	LOCAL	(none)	local-mode	no	yes	No	Specifies whether the terminal processes commands from the host or is controlled from its own keyboard
LOCK KEYBOARD	(none)	E _c K _L	locking-mode	0	0	No	Locks or unlocks the keyboard
MACRO STATUS	MACROSTATUS		macro-number		0	No	Displays the definition of a macro
MAP INDEX TO PRINT	HCMAP	E _c Q _I	monochrome-values		Error Q111	No	Specifies which graphics color indices print and which do not print on monochrome printers
MOVE	MOVE	E _c L _F	position		0,0	No	Moves the graphics position without drawing a vector
PAGE	(none)	E _c F _F				No	Erases the graphics area
PIXEL COPY	PXCOPY	E _c R _X	destination-surface		0	No	Copies pixels from one rectangular region to another
			destination-lower-left-corner		0,0		
			first-source-corner		0,0		
			second-source-corner		0,0		

^a A blank entry indicates that there is no default value.

^b You can only omit parameters in Setup.

(continued)

COMMANDS SUMMARY

Table C-1 (cont)
4100-STYLE COMMANDS

Descriptive Name	Setup Name	Host Syntax ^a	Parameters ^b	Defaults ^a		Saved	Description
				Factory	If Omitted ^b		
PROMPT MODE	PROMPTMODE	^cNM	prompt-mode	0	1	No	Starts or stops Prompt mode
RASTER WRITE	PXRASTERWRITE	^cRP	number-of-pixels		Error RP11	No	Sets the color indices individually for one or more pixels in the pixel viewport
			color-index-codes		Error RP21		
RECTANGLE FILL	PXRECTANGLE	^cRR	lower-left-corner		0,0	No	Sets all the pixels in a rectangle to the same color
			upper-right-corner		0,0		
			fill-index		0		
REPORT ERRORS	(none)	^cKQ				No	Queries the terminal for an Error Report listing the eight most recent errors
REPORT SYNTAX MODE	(none)	^c#!0				No	Queries the terminal for a Terminal Settings Report giving the current host command mode (ANSI, EDIT, TEK, or VT52)
REPORT TERMINAL SETTINGS	(none)	^cIQ	inquiry-code			No	Queries the terminal for a Terminal Settings Report
REPORT 4010 STATUS	(none)	^cEQ				No	Queries the terminal for a 4010 Status Report, listing alpha cursor position and copier status (or just GIN cursor position if 4010 GIN is enabled)
RESET	RESET	^cKV				No	Returns the terminal to its power-up condition
RUNLENGTH WRITE	PXRUNLENGTH	^cRL	runcode-array		Empty array	No	Sets one or more pixels in the pixel viewport to the same color
SAVE NONVOLATILE PARAMETERS	NVSAVE	^cKU				Yes	Saves the values of those commands whose settings can be saved in nonvolatile memory; also saves all nonvolatile macros
SELECT CODE	CODE	^c%!	syntax	0	0	Yes	Selects the host command mode, choosing ANSI, EDIT, VT52, or TEK (4100-style) command syntax
SELECT COLOR HARDCOPY IMAGE DENSITY	HCDENSITY	^cQU	density-code	1	1	Yes	Selects either fast, lower density or slow, higher density copies (4692 only)
SELECT FILL PATTERN	FILLPATTERN	^cMP	fill-pattern-number	-1	0	No	Selects a color or predefined fill pattern to fill a panel
SELECT HARDCOPY INTERFACE	HCINTERFACE	^cQD	copier-type	2	0	Yes	Selects the type of copier or printer to be used in making copies
SET ALPHA CURSOR INDICES	ACURSOR	^cTD	first-index	1	0	Yes	Assigns color indices to the alpha cursor
			second-index	1	0		
SET ALPHATEXT FONT	(none)	^c	font-code			No	Selects the font to be used for alphatext or graphtext
SET ANSWERBACK STRING	ANSWERBACK	(none)	answerback-string		Empty array	Yes	Assigns the terminal's answerback string

^a A blank entry indicates that there is no default value.

^b You can only omit parameters in Setup.

(continued)

Table C-1 (cont)
4100-STYLE COMMANDS

Descriptive Name	Setup Name	Host Syntax ^a	Parameters ^b	Defaults ^a		Saved	Description
				Factory	If Omitted ^b		
SET BAUD RATES	BAUDRATE	FcNR	transmit-data-rate	2400	Error NR11	Yes	Sets the terminal's transmit and receive data rates
			receive-data-rate	2400	Same as transmit data rate		
SET BREAK TIME	BREAKTIME	FcNK	break-time	200	0	Yes	Sets the duration (in milliseconds) of the terminal's break signal
SET BYPASS CANCEL CHARACTER	BYPASSCANCEL	FcNU	bypass-cancel-character	10(1r)	0(1c)	Yes	Specifies the character that cancels Bypass mode
SET COLOR COPIER REPAINT	HCREPAINT	FcQT	repaint-count	1	1	Yes	Specifies the number of times the 4692 Color Graphics Copier overprints each image
SET COPY SIZE	HCSIZE	FcQA	size	0	0	Yes	Selects a small image size for dialog or graphics copies (4695 only) or for dialog copies only (4691 and 4692)
SET DIALOG AREA BUFFER SIZE	DABUFFER	FcLB	number-of-lines	49	Error LB11	Yes	Specifies the number of lines available for storing text in the dialog area buffer
SET DIALOG AREA COLOR MAP	DACMAP	FcTF	color-mixture	See Table 5-4	No change	Yes	Specifies the colors assigned to color indices in the dialog area
SET DIALOG AREA HARDCOPY ATTRIBUTES	HCDAATTRIBUTES	FcQL	number-of-pages	1	No change	Yes	Sets attributes for a copy initiated by the HARDCOPY command or the D Copy key
			page-origin	0	0		
			Ff-interpretation	0	0		
SET DIALOG AREA INDEX	DAINDEX	FcLI	character-index	1	0	Yes	Specifies the color index for alphatext characters, character-cell background, and dialog area background
			character-background-index	0	0		
			dialog-background-index	0	0		
SET DIALOG AREA LINES	DALINES	FcLL	number-of-lines	30	Error LL11	Yes	Specifies the number of lines of the dialog area buffer that are displayed on the screen
SET DIALOG AREA VISIBILITY	DAVISIBILITY	FcLV	visibility-mode	1	1	Yes	Makes the dialog area visible or invisible
SET DIALOG AREA WRITING MODE	DAMODE	FcLM	writing-mode	0	0	Yes	Controls how Space and Underscore Characters are treated
SET ECHO	ECHO	FcKE	echo-mode	0	1	Yes	Specifies whether the terminal echoes characters it transmits to the host
SET EDIT CHARACTERS	EDITCHARS	FcKZ	character-delete	127(Dr)	No change	Yes	Specifies the special editing characters used in the dialog area while in Setup
			line-delete	24(Cs)	No change		
			literal	126(~)	No change		
SET EOF STRING	EOFSTRING	FcNE	EOF-string		Empty array	Yes	Specifies the terminal's end-of-file string
SET EOL STRING	EOLSTRING	FcNT	EOL-string	13(Cr)	Empty array	Yes	Specifies the terminal's end-of-line string

^a A blank entry indicates that there is no default value.

^b You can only omit parameters in Setup.

(continued)

COMMAND SUMMARY

Table C-1 (cont)
4100-STYLE COMMANDS

Descriptive Name	Setup Name	Host Syntax ^a	Parameters ^b	Defaults ^a		Saved	Description
				Factory	If Omitted ^b		
SET EOM CHARACTERS	EOMCHARS	E _c NC	first-EOM-character	13(C _R)	0(N _U)	Yes	Specifies two characters that the terminal can use to mark the end of a line of data sent to the host
			second-EOM-character	10(L _F)	0(N _U)		
SET ERROR THRESHOLD	ERRORLEVEL	E _c KT	error-threshold-level	2	0	No	Specifies the levels of error messages the terminal displays
SET FLAGGING MODE	FLAGGING	E _c NF	flagging-mode	0	0	Yes	Specifies whether the terminal uses DC1/DC3 or DTR/CTS flagging
SET GIN CURSOR COLOR	GCURSOR	E _c TC	first-color-coordinate	0	0	Yes	Specifies the color mixture for the GIN cursor
			second-color-coordinate	100	0		
			third-color-coordinate	0	0		
SET GIN CURSOR SPEED	GSPEED	E _c IJ	normal-speed	10	1	Yes	Determines how fast the GIN cursor moves across the screen when the Joydisk is pressed
			shifted-speed	1	1		
SET GRAPHICS AREA WRITING MODE	GAMODE	E _c MG	writing-mode	1	0	Yes	Specifies whether the terminal overwrites or replaces characters or markers in the graphics area
SET GRAPHTEXT CHARACTER PATH	GTPATH	E _c MN	direction	0	0	No	Selects a direction (right, left, up, down) to move after writing a graphtext character
SET GRAPHTEXT ROTATION	GTROTATION	E _c MR	mantissa	0	0	No	Specifies the rotation angle (in degrees) for graphtext
			power-of-two	0	0		
SET GRAPHTEXT SIZE	GTSIZE	E _c MC	width (unused)	Unused	Unused	No	Selects the size of graphtext
			height	61	61		
			spacing (unused)	Unused	Unused		
SET HARDCOPY MONOCHROME ATTRIBUTES	HCMONOCHROME	E _c QE	monochrome-attributes-array	1	0	Yes	Specifies the line termination (C _R or C _R L _F) that the terminal sends to a monochrome printer
SET IMAGE ORIENTATION	HCORIENT	E _c QO	orientation	0	0	Yes	Specifies whether the long axis of the image aligns with the long or short axis of the paper (4691 and 4692 only)
SET KEY EXECUTE CHARACTER	KEYEXCHAR	E _c KY	key-execute-character	16(D _L)	0(N _U)	Yes	Specifies the character used in key macros to determine whether subsequent characters will be processed by the host or by the terminal
SET LINE INDEX	LINEINDEX	E _c ML	line-index	1	0	No	Specifies the color index for all subsequent lines, panel boundaries, and markers
SET LINE STYLE	LINESTYLE	E _c MV	line-style	0	0	No	Specifies the line style for subsequent lines and panel boundaries
SET MARKER TYPE	MARKERTYPE	E _c MM	marker-number	0	0	No	Selects the kind of marker to be drawn
SET PARITY	PARITY	E _c NP	parity-mode	0	0	Yes	Specifies the kind of parity the terminal uses when it sends data to the host

^a A blank entry indicates that there is no default value.

^b You can only omit parameters in Setup.

(continued)

Table C-1 (cont)
4100-STYLE COMMANDS

Descriptive Name	Setup Name	Host Syntax ^a	Parameters ^b	Defaults ^a		Saved	Description
				Factory	If Omitted ^b		
SET PIXEL BEAM POSITION	PXPOSITION	F _C RH	beam-position	0,359	0,0	No	Sets the position of the pixel beam in the pixel viewport
SET PIXEL VIEWPORT	PXVIEWPORT	F _C RS	lower-left	0,0	0,0	No	Sets the pixel viewport's size and position in graphics memory
			upper-right	479x359	0,0		
SET PROMPT STRING	PROMPTSTRING	F _C NS	prompt-string		Empty Array	Yes	Specifies the string that initiates Prompt mode
SET QUEUE SIZE	QUEUE SIZE	F _C NQ	queue-size	300	Error NQ11	Yes	Specifies the size (in bytes) of the terminal's input queue
SET SEGMENT POSITION	SGPOSITION	F _C SX	segment-number		0	No	Moves the GIN cursor to a specified position in terminal space
			position	0.0	0.0		
SET SNOOPY MODE	SNOOPY	F _C KS	snoopy-mode	0	1	No	Specifies whether the terminal displays ASCII control characters
SET STOP BITS	STOPBITS	F _C NB	number-of-stop-bits	1	Error NB11	Yes	Specifies the number of stop bits appended to each character the terminal transmits
SET SURFACE COLOR MAP	CMAP	F _C TG	surface-number	1	Error TG11	No	Defines the color map for the graphics area
			color-mixtures	See Table 5-7	No change	No	
SET TAB STOPS	TABS	F _C KB	tab-positions	Every eighth column (1,9,17,...)	0	Yes	Sets tab stops at the specified positions
SET TEXT INDEX	GTINDEX	F _C MT	text-index	1	0	No	Specifies the color index for all text displayed in the graphics area
SET TRANSMIT DELAY	XMTDELAY	F _C ND	transmit-delay	100	0	Yes	Specifies the terminal's delay (in milliseconds) between transmitting lines of text
SET TRANSMIT RATE LIMIT	XMTLIMIT	F _C NL	rate-limit	19200	Error NL11	Yes	Paces the terminal's data transmission so that it doesn't exceed the indicated rate
SET VIEW ATTRIBUTES	VIEWATTRIBUTE	F _C RA	surface-number	1	0	No	Selects the background index for the graphics area
			wipe-index	0	0		
			border-index	1	0		
SET WINDOW	WINDOW	F _C RW	first-corner	0,0	0,0	No	Sets the boundaries of the current window in terminal space
			second-corner	4095,3132	0,0		
SET 4014 LINE STYLE	(none)	F _C	line-style-code			No	Specifies line styles compatible with Tektronix 4014, 4016, and 4114 terminals
STATUS	STATUS	(none)	name		All commands	No	Displays the current settings for a command or a cluster of commands
4010 HARDCOPY	(none)	F _C F _B				No	Generates a hard copy of the entire screen

^a A blank entry indicates that there is no default value.

^b You can only omit parameters in Setup.

ANSI-STYLE COMMANDS

Table C-2 lists the commands that are available to the host computer in ANSI mode and EDIT mode. The table has the same format as Table C-1, with a few differences. Since many ANSI mode commands require a *terminator character* in host syntax, we've used an underscore in the *Host Syntax* column to indicate the position for the parameter entry.

For example, the syntax for the CUU (Cursor Up) command is shown as $E_c[_A]$. The parameter to enter in place of the underscore is *number-of-lines*. To move the cursor up two lines the host must send:

$E_c[2A]$

Some ANSI mode commands use two parameters in host syntax. Separate the two parameters with a semicolon (;). For example the syntax for the CUP (Cursor Position) command is $E_c[_H]$. The two parameters are *row-number* and *column-number* — so, to move the cursor to Row 5, Column 10, the host must send:

$E_c[5;10H]$

The RM (Reset Modes) and SM (Set Modes) commands have a unique format. To set or reset a mode with either of these commands, use the table this way:

- For host syntax, use the entry in the *Host Syntax* column. For these commands, the syntax in this column includes the parameter value.
- For Setup syntax, use the Setup name in the *Setup Syntax* column and the keyword (shown in all uppercase characters) in the *Parameters* column.

For example, under the RM command, the host syntax for setting Insert/Replace mode (IRM) is:

$E_c[4I]$

The 4 is the actual parameter value to enter.

The same command setting in Setup syntax is:

INSERTREPLACE REPLACE

Table C-2
ANSI-STYLE COMMANDS

Descriptive Name	Setup Name	Host Syntax ^a	Parameters ^b	Defaults ^c		Saved	Description
				Factory	If Omitted ^d		
BEL (BELL)	(none)	B _L				No	Sounds the terminal's bell
BS (BACKSPACE)	(none)	B _S				No	Moves the cursor left one column
CBT (CURSOR BACKWARD TAB)	(none)	$E_c[_Z]$	number-of-tab-stops		1	No	Moves the cursor backward to a preceding tab stop on the current line
CHT (CURSOR HORIZONTAL TAB)	(none)	$E_c[_I]$	number-of-following-tab-stops		1	No	Moves the cursor to a following tab stop on the current line
CAN (CANCEL)	(none)	C _N				No	Cancels an ANSI-mode command in progress
CPR (CURSOR POSITION REPORT)	(none)	$E_c[_R]$	row column			No	A report sent by the terminal to the host in response to a DSR command to provide the row and column address of the cursor
CR (CARRIAGE RETURN)	(none)	C _R				No	Moves the cursor to the first column of the current line
CUB (CURSOR BACKWARD)	(none)	$E_c[_D]$	number-of-columns		1	No	Moves the cursor left one or more columns
CUD (CURSOR DOWN)	(none)	$E_c[_B]$	number-of-lines		1	No	Moves the cursor down one or more lines
CUF (CURSOR FORWARD)	(none)	$E_c[_C]$	number-of-columns		1	No	Moves the cursor one or more columns to the right

^a In ANSI mode host syntax, most commands require a terminator character following any parameter value. An underscore _ shows where to place the parameter. (continued)

^b A capitalized keyword in this column indicates the only valid entry for the command.

^c A blank entry indicates that there is no default value.

^d You can only omit parameters in Setup.

Table C-2 (cont)
ANSI-STYLE COMMANDS

Descriptive Name	Setup Name	Host Syntax ^a	Parameters ^b	Defaults ^c		Saved	Description
				Factory	If Omitted ^d		
CUP (CURSOR POSITION)	(none)	E _c [_H]	row-number		1	No	Moves the cursor to the specified row and column
			column-number		1		
CUU (CURSOR UP)	(none)	E _c [_A]	number-of-lines		1	No	Moves the cursor upward one or more lines
DA (DEVICE ATTRIBUTES)	(none)	E _c [_c]	device-status-request		0	No	Tells the terminal to report what kind of a terminal it is
DCH (DELETE CHARACTER)	(none)	E _c [_P]	number-of-characters		1	No	Deletes one or more characters, starting at the cursor position
DL (DELETE LINE)	(none)	E _c [_M]	number-of-lines		1	No	Deletes one or more lines, starting with the current line
DMI (DISABLE MANUAL INPUT)	(none)	E _c ^I				No	Disables the keyboard
DSR (DEVICE STATUS REPORT)	(none)	E _c [_n]	status			No	Queries the terminal for a CPR (Cursor Position Report) or a DSR (Device Status Report)
ECH (ERASE CHARACTER)	(none)	E _c [_X]	number-of-characters		1	No	Erases one or more characters, starting at the cursor position
ED (ERASE IN DISPLAY)	(none)	E _c [_J]	erase-extent		0	No	Erases all or part of the dialog buffer
EL (ERASE IN LINE)	(none)	E _c [_K]	erase-extent		0	No	Erases all or part of the current line
EMI (ENABLE MANUAL INPUT)	(none)	E _c ^b				No	Enables the keyboard
ENQUIRY	(none)	E _c ^Q				No	Queries the terminal for its answerback string
FF (FORM FEED)	(none)	E _c ^F				No	Inserts E _c ^F into the dialog area and advances the cursor position
HT (HORIZONTAL TAB)	(none)	E _c ^T				No	Advances the cursor to the next horizontal tab stop on the current line
HTS (HORIZONTAL TAB SET)	(none)	E _c ^H				No	Sets a tab stop at the current cursor location
	TABS	(none)	list-of-tab-stops			No	Sets tab stops at the specified locations
HVP (HORIZONTAL AND VERTICAL POSITION)	(none)	E _c [_f]	row-number		1	No	Moves the cursor to a specified row and column of the screen
			column-number				
ICH (INSERT CHARACTER)	(none)	E _c [_@]	number-of-characters		1	No	Inserts one or more E _c ^P (Space) characters at the cursor position
IL (INSERT LINE)	(none)	E _c [_L]	number-of-lines		1	No	Inserts one or more blank lines in front of the current line
IND (INDEX)	(none)	E _c ^D				No	Moves the cursor down one line without moving its character position on the line

^a In ANSI mode host syntax, most commands require a terminator character following any parameter value. An underscore _ shows where to place the parameter. (continued)
^b A capitalized keyword in this column indicates the only valid entry for the command.
^c A blank entry indicates that there is no default value.
^d You can only omit parameters in Setup.

COMMAND SUMMARY

**Table C-2 (cont)
ANSI-STYLE COMMANDS**

Descriptive Name	Setup Name	Host Syntax ^a	Parameters ^b	Defaults ^c		Saved	Description
				Factory	If Omitted ^d		
IRM (INSERT/REPLACE MODE)	INSERTREPLACE	See SM and RM	mode	replace	replace	No	Specifies whether each newly entered character overwrites existing characters or is inserted at the cursor position
LF (LINE FEED)	(none)	^L F				No	Moves the cursor down one line
LNМ (LINEFEED/NEWLINE MODE)	LFCR	See SM and RM	mode	no	yes	Yes	Specifies whether a ^L F (Line Feed) character sent to the terminal also implies a ^C R (Carriage Return)
MC (MEDIA COPY)	(none)	^E c[_i]	copy-option		0	No	Sends dialog to the printer at the same time as it is written to the screen
NEL (NEW LINE)	(none)	^E cE				No	Moves the cursor to the start of the next line
REPORT SYNTAX MODE	(none)	^E c#!0				No	Queries the terminal for a Terminal Settings Report giving the terminal's host command mode (ANSI, EDIT, TEK, or VT52)
RI (REVERSE INDEX)	(none)	^E cM				No	Moves the cursor up one line without affecting the cursor position on the line
RIS (RESET TO INITIAL STATE)	(none)	^E cc				No	Resets certain terminal attributes to their power-up values
RM (RESET MODE)	(none)	^E c[_l]	mode	(none)	Error [l11	Yes	Resets one or more modes
IRM (INSERT/REPLACE MODE)	INSERTREPLACE	^E c[4l]	REPLACE	replace	replace	No	Specifies that each character entered overwrites (that is, replaces) the character at the cursor position
KAM (KEYBOARD ACTION MODE)	(none)	^E c[2l]		no	no	No	Enables the terminal keyboard
LNМ (LINEFEED/NEWLINE MODE)	LFCR	^E c[20l]	NO	no	yes	Yes	Specifies that ^L F does not also imply ^C R
SRM (SEND/RECEIVE MODE)	ECHO	^E c[12l]	YES	no	yes	Yes	Specifies that the terminal displays characters as they are sent to the host (local echo)
TEKANM (ANSI-TO-VT52 MODE)	CODE	^E c[?2l]	VT52	TEK	TEK	Yes	Puts the terminal in VT52 mode
TEKARM (AUTOREPEAT MODE)	AUTOREPEAT	^E c[?8l]	NO	yes	yes	Yes	Disables autorepeat so that keyboard keys do not repeat when held down
TEKAWM (AUTOWRAP MODE)	AUTOWRAP	^E c[?7l]	NO	yes	yes	Yes	Disables autowrap, so that characters entered in the rightmost column write over existing characters in that column

^a In ANSI mode host syntax, most commands require a terminator character following any parameter value. An underscore _ shows where to place the parameter. (continued)

^b A capitalized keyword in this column indicates the only valid entry for the command.

^c A blank entry indicates that there is no default value.

^d You can only omit parameters in Setup.

Table C-2 (cont)
ANSI-STYLE COMMANDS

Descriptive Name	Setup Name	Host Syntax ^a	Parameters ^b	Defaults ^c		Saved	Description
				Factory	If Omitted ^d		
RM (RESET MODE) (cont)							
TEKCKM (CURSOR KEYS MODE)	CURSORKYMODE	Ec[?11	NO	no	yes	No	Specifies that Function Keys F1 — F4 send ANSI cursor commands (or send programmed values, if key expansion is enabled)
TEKCOLM (COLUMN MODE)	COLUMNMODE	Ec[?31	80	80	80	Yes	Specifies a dialog area buffer width of 80 columns
TEKOM (ORIGIN MODE)	ORIGINMODE	Ec[?61	ABSOLUTE	relative	relative	Yes	Selects Origin mode Absolute; cursor moves to Row 1, Column 1 of dialog area buffer
TEKORM (OVERSTRIKE/REPLACE MODE)	DAMODE	Ec[< 11	OVERSTRIKE	replace	replace	No	Causes ^S P (Space) and <u> Underscore to be treated as normal characters</u>
TEKSCNM (SCREEN MODE)	SCREENMODE	Ec[?51	NORMAL	normal	normal	Yes	Specifies that color mixtures have their normal hues and that Index 0 of the dialog area is transparent
SCS (SELECT CHARACTER SET)	SELECTCHARSET G0	Ec(character-set	depends on keyboard	Error	No	Selects one or two character sets from the eight stored in the terminal's firmware and makes them available through the keyboard
	SELECTCHARSET G1	Ec)	character-set	depends on keyboard	Error		
SD (SCROLL DOWN)	(none)	Ec[_T	number-of-lines		1	No	Scrolls lines down
SELECT CODE	CODE	Ec%!	syntax		0 (TEK mode)	Yes	Selects the host command mode, choosing ANSI, EDIT, VT52, or TEK (4100-style) command syntax
SGR (SELECT GRAPHIC RENDITION)	TEXTRENDITION	Ec[_m	graphic-rendition		0	No	Selects display attributes for the dialog area
SI (SHIFT IN)	(none)	S ₁				No	Invokes the current G0 character set
SL (SHIFT LEFT)	(none)	Ec[_ ^S P@	number-of-columns		1	No	Moves the visible columns of the dialog area to the left
SM (SET MODES)	(none)	Ec[_h	mode			Yes	Sets one or more modes
IRM (INSERT/REPLACE MODE)	INSERTREPLACE	Ec[4h	INSERT	replace	replace	No	Specifies that each character is inserted at the cursor position
KAM (KEYBOARD ACTION MODE)	(none)	Ec[2h				No	Disables the terminal keyboard
LNM (LINEFEED/NEWLINE MODE)	LFCR	Ec[20h	YES	no	yes	Yes	Specifies that ^L F also implies ^C R

^a In ANSI mode host syntax, most commands require a terminator character following any parameter value. An underscore shows where to place the parameter. (continued)
^b A capitalized keyword in this column indicates the only valid entry for the command.
^c A blank entry indicates that there is no default value.
^d You can only omit parameters in Setup.

COMMAND SUMMARY

Table C-2 (cont)
ANSI-STYLE COMMANDS

Descriptive Name	Setup Name	Host Syntax ^a	Parameters ^b	Defaults ^c		Saved	Description
				Factory	If Omitted ^d		
SM (SET MODES) (cont)							
SRM (SEND/RECEIVE MODE)	ECHO	$E_c[12h$	NO	no	yes	Yes	Specifies that the terminal does not display characters as they are sent to the host
TEKARM (AUTOREPEAT MODE)	AUTOREPEAT	$E_c[?8h$	YES	yes	yes	Yes	Enables autorepeat so that keyboard keys repeat when held down
TEKAWM (AUTOWRAP MODE)	AUTOWRAP	$E_c[?7h$	NO	yes	yes	Yes	Enables autowrap, so that characters entered in the rightmost column wrap around to next line
TEKCKM (CURSOR KEYS MODE)	CURSORKEYMODE	$E_c[?h$	YES	no	yes	No	Specifies that Function Keys F1 — F4 transmit application program codes
TEKCOLM (COLUMN MODE)	COLUMNMODE	$E_c[?3h$	132	80	80	Yes	Specifies a dialog area buffer width of 132 columns
TEKOM (ORIGIN MDOE)	ORIGINMODE	$E_c[?6h$	RELATIVE	relative	relative	Yes	Selects Origin mode Relative; cursor moves to Row 1, Column 1 of scrolling region
TEKORM (OVERSTRIKE/REPLACE MODE)	DAMODE	$E_c[<1h$	REPLACE	replace	replace	No	Specifies that <u> underscores the current character and that S_p moves the cursor without erasing the current character</u>
TEKSCNM (SCREEN MODE)	SCREENMODE	$E_c[?5h$	REVERSE	normal	normal	Yes	Reverses the hues of color mixtures and makes Index 0 of the dialog area opaque
SO (SHIFT OUT)	(none)	S_o				No	Invokes the G0 character set
SR (SCROLL RIGHT)	(none)	$E_c[_S pA$	number-of-columns		1	No	Moves the visible columns of the dialog area to the right
SRM (SEND/RECEIVE MODE)	ECHO	See SM and RM	mode	no	yes	Yes	Specifies whether the terminal provides its own echo (local echo) of data entered at the keyboard
SU (SCROLL UP)	(none)	$E_c[_S$	number-of-lines		1	No	Scrolls lines up
SUB (SUBSTITUTE)	(none)	S_b					Cancels an ANSI mode command in progress
TBC (TAB CLEAR)	(none)	$E_c[_g$	tab-clear-extent		0	No	Clears one or more tab stops
TEKANM (ANSI-TO-VT52 MODE)	CODE VT52	See SM and RM				Yes	Selects VT52 mode
TEKARM (AUTOREPEAT MODE)	AUTOREPEAT	See SM and RM	mode	yes	yes	Yes	Specifies whether keyboard keys repeat when held down
TEKAWM (AUTOWRAP MODE)	AUTOWRAP	See SM and RM	mode	yes	yes	Yes	Specifies whether characters written to the rightmost column overwrite existing characters or wrap to the next line

^a In ANSI mode host syntax, most commands require a terminator character following any parameter value. An underscore shows where to place the parameter. (continued)

^b A capitalized keyword in this column indicates the only valid entry for the command.

^c A blank entry indicates that there is no default value.

^d You can only omit parameters in Setup.

Table C-2 (cont)
ANSI-STYLE COMMANDS

Descriptive Name	Setup Name	Host Syntax ^a	Parameters ^b	Defaults ^c		Saved	Description
				Factory	If Omitted ^d		
TEKCKM (CURSOR KEYS MODE)	CURSORKYMODE	See SM and RM	mode	no	yes	No	Specifies whether Function Keys F1 — F4 transmit ANSI cursor control commands
TEKCOLM (COLUMN MODE)	COLUMNMODE	See SM and RM	mode	80	80	Yes	Selects 80- or 132-column width for the dialog area buffer
TEKDHL (DOUBLE HEIGHT LINE)	(none)	E _c #3 (top)				No	Causes the current line to become the top or bottom half of a double-height, double-width line
	(none)	E _c #4 (bottom)					
TEKDWL (DOUBLE WIDTH LINE)	(none)	E _c #6				No	Causes the current line to become a double-width, single-height line
TEKID (IDENTIFY TERMINAL)	(none)	E _c Z				No	Tells the terminal to report what kind of terminal it is
TEKPPAM (KEYPAD APPLICATION MODE)	KEYPADMODE	E _c =	keypad-mode			No	Causes the numeric keypad to send different characters than the numeric keys on the main keyboard
TEKPPNM (KEYPAD NUMERIC MODE)	KEYPADMODE	E _c >	keypad-mode			No	Causes the numeric keypad to send the same characters as the numeric keys on the main keyboard
TEKOM (ORIGIN MODE)	ORIGINMODE	See SM and RM	mode	relative	relative	Yes	Specifies how the terminal interprets cursor addresses in ANSI commands
TEKORM (OVERSTRIKE/REPLACE MODE)	DAMODE	See SM and RM	mode	replace	replace	Yes	Controls how the Space and Underscore characters are treated
TEKRC (RESTORE CURSOR)	(none)	E _c 8				No	Restores the cursor position, graphic rendition, character set, and Origin mode previously saved using the TEKSC (Save Cursor) command
TEKSC (SAVE CURSOR)	(none)	E _c 7				No	Saves the cursor position, graphic rendition, character set, and Origin mode
TEKSCNM (SCREEN MODE)	SCREENMODE	See SM and RM	mode	normal	reverse	Yes	Displays colors in the dialog area in normal or reversed values
TEKSTBM (SET TOP AND BOTTOM MARGINS)	EDITMARGIN	E _c _r	top-margin	1	1	No	Sets the dialog area buffer's edit margins
			bottom-margin	30	Same as DALINES		
TEKSWL (SINGLE WIDTH LINE)	(none)	E _c #5				No	Causes the current line to become a single-width, single-height line
V _T (VERTICAL TAB)	(none)	V _T				No	Moves the cursor down one line without affecting the cursor position on the line

^a In ANSI mode host syntax, most commands require a terminator character following any parameter value. An underscore _ shows where to place the parameter.

^b A capitalized keyword in this column indicates the only valid entry for the command.

^c A blank entry indicates that there is no default value.

^d You can only omit parameters in Setup.

COMMAND SUMMARY

VT52-STYLE COMMANDS

Table C-3 lists the commands that are available to the host computer in VT52 mode. The table has the same format as Table C-1

Table C-3
VT52-STYLE COMMANDS

Descriptive Name	Setup Name	Host Syntax	Parameters	Defaults ^a		Description
				Factory	If Omitted ^b	
CURSOR DOWN	(none)	E _c B				Moves the cursor down one line without moving it horizontally
CURSOR LEFT	(none)	E _c D				Moves the cursor one column to the left
CURSOR RIGHT	(none)	E _c C				Moves the cursor one column to the right
CURSOR TO HOME	(none)	E _c H				Moves the cursor to the home position: Row 1, Column 1
CURSOR UP	(none)	E _c A				Moves the cursor up one line without moving it horizontally
DIRECT CURSOR ADDRESS	(none)	E _c Y	line column			Moves the cursor to the specified line and column
ENQUIRY	(none)	E _c Q				Queries the terminal for its answerback string
ENTER ALTERNATE KEYPAD MODE	KEYPADMODE APPLICATION	E _c =				Causes the numeric keypad keys and Function Keys F5—F8 to assume their Alternate Keypad mode meanings (shown in Table 3-12)
ENTER ANSI MODE	CODE ANSI	E _c				Places the terminal in ANSI mode
ENTER GRAPHICS MODE	(none)	E _c F				Selects the Rulings character set as the G0 character set
ERASE TO END OF LINE	(none)	E _c K				Erases from the cursor to the end of the line
ERASE TO END OF SCREEN	(none)	E _c J				Erases from the cursor to the end of the screen
EXIT ALTERNATE KEYPAD MODE	KEYPADMODE NUMERIC	E _c >				Causes the numeric keypad keys and the Function keys F5—F8 to assume their factory default meanings, or their programmed meanings if they have been programmed
EXIT GRAPHICS MODE	(none)	E _c G				Restores the G0 character set that was in effect before the current ENTER GRAPHICS MODE command was issued
IDENTIFY	(none)	E _c Z				Identifies the terminal to the host
REPORT SYNTAX MODE	(none)	E _c #!0				Sends a Terminal Status Report that contains the syntax mode status to the host
REVERSE LINEFEED	(none)	E _c I				Moves the cursor up one line without affecting the cursor position on the line
SELECT CODE	CODE	E _c %!	syntax		0 (TEK)	Causes the terminal to recognize ANSI, EDIT, TEK, or VT52 command syntax

^a A blank entry indicates that there is no default value.

^b You can only omit parameters in Setup.

Appendix D

ENCODED INTEGER PARAMETERS

Table D-1 lists integer parameters between -1199 and +1199.

Table D-1
ENCODED INTEGER PARAMETERS

Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param
0	0	-0	s _p	50	C2	-50	C"	100	F4	-100	F\$	150	I6	-150	I&
1	1	-1	!	51	C3	-51	C#	101	F5	-101	F%	151	I7	-151	I'
2	2	-2	"	52	C4	-52	C\$	102	F6	-102	F&	152	I8	-152	I(
3	3	-3	#	53	C5	-53	C%	103	F7	-103	F'	153	I9	-153	I)
4	4	-4	\$	54	C6	-54	C&	104	F8	-104	F(154	I:	-154	I*
5	5	-5	%	55	C7	-55	C'	105	F9	-105	F)	155	I;	-155	I+
6	6	-6	&	56	C8	-56	C(106	F:	-106	F*	156	I<	-156	I,
7	7	-7	'	57	C9	-57	C)	107	F;	-107	F+	157	I=	-157	I-
8	8	-8	(58	C:	-58	C*	108	F<	-108	F,	158	I>	-158	I.
9	9	-9)	59	C;	-59	C+	109	F=	-109	F-	159	I?	-159	I/
10	:	-10	*	60	C<	-60	C,	110	F>	-110	F.	160	J0	-160	J ^{s_p}
11	;	-11	+	61	C=	-61	C-	111	F?	-111	F/	161	J1	-161	J!
12	<	-12	,	62	C>	-62	C.	112	G0	-112	G ^{s_p}	162	J2	-162	J"
13	=	-13	-	63	C?	-63	C/	113	G1	-113	G!	163	J3	-163	J#
14	>	-14	.	64	D0	-64	D ^{s_p}	114	G2	-114	G"	164	J4	-164	J\$
15	?	-15	/	65	D1	-65	D'	115	G3	-115	G#	165	J5	-165	J%
16	A0	-16	A ^{s_p}	66	D2	-66	D"	116	G4	-116	G\$	166	J6	-166	J&
17	A1	-17	A!	67	D3	-67	D#	117	G5	-117	G%	167	J7	-167	J'
18	A2	-18	A"	68	D4	-68	D\$	118	G6	-118	G&	168	J8	-168	J(
19	A3	-19	A#	69	D5	-69	D%	119	G7	-119	G'	169	J9	-169	J)
20	A4	-20	A\$	70	D6	-70	D&	120	G8	-120	G(170	J:	-170	J*
21	A5	-21	A%	71	D7	-71	D'	121	G9	-121	G)	171	J;	-171	J+
22	A6	-22	A&	72	D8	-72	D(122	G:	-122	G*	172	J<	-172	J,
23	A7	-23	A'	73	D9	-73	D)	123	G;	-123	G+	173	J=	-173	J-
24	A8	-24	A(74	D:	-74	D*	124	G<	-124	G,	174	J>	-174	J.
25	A9	-25	A)	75	D;	-75	D+	125	G=	-125	G-	175	J?	-175	J/
26	A:	-26	A*	76	D<	-76	D,	126	G>	-126	G.	176	K0	-176	K ^{s_p}
27	A;	-27	A+	77	D=	-77	D-	127	G?	-127	G/	177	K1	-177	K!
28	A<	-28	A,	78	D>	-78	D.	128	H0	-128	H ^{s_p}	178	K2	-178	K"
29	A=	-29	A-	79	D?	-79	D/	129	H1	-129	H!	179	K3	-179	K#
30	A>	-30	A.	80	E0	-80	E ^{s_p}	130	H2	-130	H"	180	K4	-180	K\$
31	A?	-31	A/	81	E1	-81	E!	131	H3	-131	H#	181	K5	-181	K%
32	B0	-32	B ^{s_p}	82	E2	-82	E"	132	H4	-132	H\$	182	K6	-182	K&
33	B1	-33	B!	83	E3	-83	E#	133	H5	-133	H%	183	K7	-183	K'
34	B2	-34	B"	84	E4	-84	E\$	134	H6	-134	H&	184	K8	-184	K(
35	B3	-35	B#	85	E5	-85	E%	135	H7	-135	H'	185	K9	-185	K)
36	B4	-36	B\$	86	E6	-86	E&	136	H8	-136	H(186	K:	-186	K*
37	B5	-37	B%	87	E7	-87	E'	137	H9	-137	H)	187	K;	-187	K+
38	B6	-38	B&	88	E8	-88	E(138	H:	-138	H*	188	K<	-188	K,
39	B7	-39	B'	89	E9	-89	E)	139	H;	-139	H+	189	K=	-189	K-
40	B8	-40	B(90	E:	-90	E*	140	H<	-140	H,	190	K>	-190	K.
41	B9	-41	B)	91	E;	-91	E+	141	H=	-141	H-	191	K?	-191	K/
42	B:	-42	B*	92	E<	-92	E,	142	H>	-142	H.	192	L0	-192	L ^{s_p}
43	B;	-43	B+	93	E=	-93	E-	143	H?	-143	H/	193	L1	-193	L!
44	B<	-44	B,	94	E>	-94	E.	144	I0	-144	I ^{s_p}	194	L2	-194	L"
45	B=	-45	B-	95	E?	-95	E/	145	I1	-145	I!	195	L3	-195	L#
46	B>	-46	B.	96	F0	-96	F ^{s_p}	146	I2	-146	I"	196	L4	-196	L\$
47	B?	-47	B/	97	F1	-97	F!	147	I3	-147	I#	197	L5	-197	L%
48	C0	-48	C ^{s_p}	98	F2	-98	F"	148	I4	-148	I\$	198	L6	-198	L&
49	C1	-49	C!	99	F3	-99	F#	149	I5	-149	I%	199	L7	-199	L'

^a Positive integer
^b Negative integer

(continued)

ENCODED INTEGER PARAMETERS

Table D-1 (cont)

ENCODED INTEGER PARAMETERS

Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param
200	L8	-200	L (250	O:	-250	O *	300	R<	-300	R,	350	U>	-350	U.
201	L9	-201	L)	251	O;	-251	O +	301	R=	-301	R-	351	U?	-351	U /
202	L:	-202	L *	252	O<	-252	O,	302	R>	-302	R.	352	V0	-352	V ^{Sp}
203	L;	-203	L+	253	O=	-253	O-	303	R?	-303	R /	353	V1	-353	V!
204	L<	-204	L,	254	O>	-254	O.	304	S0	-304	S ^{Sp}	354	V2	-354	V"
205	L=	-205	L-	255	O?	-255	O /	305	S1	-305	S!	355	V3	-355	V#
206	L>	-206	L.	256	P0	-256	P ^{Sp}	306	S2	-306	S"	356	V4	-356	V\$
207	L?	-207	L /	257	P1	-257	P!	307	S3	-307	S#	357	V5	-357	V%
208	M0	-208	M ^{Sp}	258	P2	-258	P"	308	S4	-308	S\$	358	V6	-358	V&
209	M1	-209	M!	259	P3	-259	P#	309	S5	-309	S%	359	V7	-359	V /
210	M2	-210	M"	260	P4	-260	P\$	310	S6	-310	S&	360	V8	-360	V (
211	M3	-211	M#	261	P5	-261	P%	311	S7	-311	S /	361	V9	-361	V)
212	M4	-212	M\$	262	P6	-262	P&	312	S8	-312	S (362	V:	-362	V *
213	M5	-213	M%	263	P7	-263	P /	313	S9	-313	S)	363	V;	-363	V +
214	M6	-214	M&	264	P8	-264	P (314	S:	-314	S *	364	V<	-364	V,
215	M7	-215	M /	265	P9	-265	P)	315	S;	-315	S +	365	V=	-365	V-
216	M8	-216	M (266	P:	-266	P *	316	S<	-316	S,	366	V>	-366	V.
217	M9	-217	M)	267	P;	-267	P +	317	S=	-317	S-	367	V?	-367	V /
218	M:	-218	M *	268	P<	-268	P,	318	S>	-318	S.	368	W0	-368	W ^{Sp}
219	M;	-219	M +	269	P=	-269	P-	319	S?	-319	S /	369	W1	-369	W!
220	M<	-220	M,	270	P>	-270	P.	320	T0	-320	T ^{Sp}	370	W2	-370	W"
221	M=	-221	M-	271	P?	-271	P /	321	T1	-321	T!	371	W3	-371	W#
222	M>	-222	M.	272	Q0	-272	Q ^{Sp}	322	T2	-322	T"	372	W4	-372	W\$
223	M?	-223	M /	273	Q1	-273	Q!	323	T3	-323	T#	373	W5	-373	W%
224	N0	-224	N ^{Sp}	274	Q2	-274	Q"	324	T4	-324	T\$	374	W6	-374	W&
225	N1	-225	N!	275	Q3	-275	Q#	325	T5	-325	T%	375	W7	-375	W /
226	N2	-226	N"	276	Q4	-276	Q\$	326	T6	-326	T&	376	W8	-376	W (
227	N3	-227	N#	277	Q5	-277	Q%	327	T7	-327	T /	377	W9	-377	W)
228	N4	-228	N\$	278	Q6	-278	Q&	328	T8	-328	T (378	W:	-378	W *
229	N5	-229	N%	279	Q7	-279	Q /	329	T9	-329	T)	379	W;	-379	W +
230	N6	-230	N&	280	Q8	-280	Q (330	T:	-330	T *	380	W<	-380	W,
231	N7	-231	N /	281	Q9	-281	Q)	331	T;	-331	T +	381	W=	-381	W-
232	N8	-232	N (282	Q:	-282	Q *	332	T<	-332	T,	382	W>	-382	W.
233	N9	-233	N)	283	Q;	-283	Q +	333	T=	-333	T-	383	W?	-383	W /
234	N:	-234	N *	284	Q<	-284	Q,	334	T>	-334	T.	384	X0	-384	X ^{Sp}
235	N;	-235	N +	285	Q=	-285	Q-	335	T?	-335	T /	385	X1	-385	X!
236	N<	-236	N,	286	Q>	-286	Q.	336	U0	-336	U ^{Sp}	386	X2	-386	X"
237	N=	-237	N-	287	Q?	-287	Q /	337	U1	-337	U!	387	X3	-387	X#
238	N>	-238	N.	288	R0	-288	R ^{Sp}	338	U2	-338	U"	388	X4	-388	X\$
239	N?	-239	N /	289	R1	-289	R!	339	U3	-339	U#	389	X5	-389	X%
240	O0	-240	O ^{Sp}	290	R2	-290	R"	340	U4	-340	U\$	390	X6	-390	X&
241	O1	-241	O!	291	R3	-291	R#	341	U5	-341	U%	391	X7	-391	X /
242	O2	-242	O"	292	R4	-292	R\$	342	U6	-342	U&	392	X8	-392	X (
243	O3	-243	O#	293	R5	-293	R%	343	U7	-343	U /	393	X9	-393	X)
244	O4	-244	O\$	294	R6	-294	R&	344	U8	-344	U (394	X:	-394	X *
245	O5	-245	O%	295	R7	-295	R /	345	U9	-345	U)	395	X;	-395	X +
246	O6	-246	O&	296	R8	-296	R (346	U:	-346	U *	396	X<	-396	X,
247	O7	-247	O /	297	R9	-297	R)	347	U;	-347	U +	397	X=	-397	X-
248	O8	-248	O (298	R:	-298	R *	348	U<	-348	U,	398	X>	-398	X.
249	O9	-249	O)	299	R;	-299	R +	349	U=	-349	U-	399	X?	-399	X /

^a Positive integer
^b Negative integer

(continued)

Table D-1 (cont)
ENCODED INTEGER PARAMETERS

Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param
400	Y0	-400	Y ^{Sp}	450	\2	-450	\"	500	_4	-500	__\$	550	b6	-550	b&
401	Y1	-401	Y!	451	\3	-451	\#	501	_5	-501	_%	551	b7	-551	b'
402	Y2	-402	Y"	452	\4	-452	\\$	502	_6	-502	_&	552	b8	-552	b(
403	Y3	-403	Y#	453	\5	-453	\%	503	_7	-503	_'	553	b9	-553	b)
404	Y4	-404	Y\$	454	\6	-454	\&	504	_8	-504	_ (554	b:	-554	b*
405	Y5	-405	Y%	455	\7	-455	\'	505	_9	-505	_)	555	b;	-555	b+
406	Y6	-406	Y&	456	\8	-456	\ (506	_:	-506	_*	556	b<	-556	b,
407	Y7	-407	Y'	457	\9	-457	\)	507	_;	-507	_+	557	b=	-557	b-
408	Y8	-408	Y(458	\:	-458	*	508	_<	-508	_,	558	b>	-558	b.
409	Y9	-409	Y)	459	\;	-459	\+	509	_=	-509	_-	559	b?	-559	b/
410	Y:	-410	Y*	460	\<	-460	\,	510	_>	-510	_.	560	c0	-560	c ^{Sp}
411	Y;	-411	Y+	461	\=	-461	\-	511	_?	-511	_ /	561	c1	-561	c!
412	Y<	-412	Y,	462	\>	-462	\.	512	\0	-512	\ ^{Sp}	562	c2	-562	c"
413	Y=	-413	Y-	463	\?	-463	\ /	513	\1	-513	\!	563	c3	-563	c#
414	Y>	-414	Y.	464]0	-464] ^{Sp}	514	\2	-514	\"	564	c4	-564	c\$
415	Y?	-415	Y/	465]1	-465]!	515	\3	-515	\#	565	c5	-565	c%
416	Z0	-416	Z ^{Sp}	466]2	-466]'	516	\4	-516	\\$	566	c6	-566	c&
417	Z1	-417	Z!	467]3	-467]#	517	\5	-517	\%	567	c7	-567	c'
418	Z2	-418	Z"	468]4	-468]\$	518	\6	-518	\&	568	c8	-568	c(
419	Z3	-419	Z#	469]5	-469]%	519	\7	-519	\'	569	c9	-569	c)
420	Z4	-420	Z\$	470]6	-470]&	520	\8	-520	\ (570	c:	-570	c*
421	Z5	-421	Z%	471]7	-471]'	521	\9	-521	\)	571	c;	-571	c+
422	Z6	-422	Z&	472]8	-472] (522	\:	-522	*	572	c<	-572	c,
423	Z7	-423	Z'	473]9	-473])	523	\;	-523	\+	573	c=	-573	c-
424	Z8	-424	Z(474]:	-474]*	524	\<	-524	\,	574	c>	-574	c.
425	Z9	-425	Z)	475];	-475] +	525	\=	-525	\-	575	c?	-575	c/
426	Z:	-426	Z*	476]<	-476] ,	526	\>	-526	\.	576	d0	-576	d ^{Sp}
427	Z;	-427	Z+	477] =	-477] -	527	\?	-527	\ /	577	d1	-577	d!
428	Z<	-428	Z,	478] >	-478] .	528	a0	-528	a ^{Sp}	578	d2	-578	d"
429	Z=	-429	Z-	479] ?	-479] /	529	a1	-529	a!	579	d3	-579	d#
430	Z>	-430	Z.	480	^0	-480	^ ^{Sp}	530	a2	-530	a"	580	d4	-580	d\$
431	Z?	-431	Z/	481	^1	-481	^!	531	a3	-531	a#	581	d5	-581	d%
432	[0	-432	[^{Sp}	482	^2	-482	^"	532	a4	-532	a\$	582	d6	-582	d&
433	[1	-433	[!	483	^3	-483	^#	533	a5	-533	a%	583	d7	-583	d'
434	[2	-434	["	484	^4	-484	^\$	534	a6	-534	a&	584	d8	-584	d(
435	[3	-435	[#	485	^5	-485	^%	535	a7	-535	a'	585	d9	-585	d)
436	[4	-436	[\$	486	^6	-486	^&	536	a8	-536	a(586	d:	-586	d*
437	[5	-437	[%	487	^7	-487	^'	537	a9	-537	a)	587	d;	-587	d+
438	[6	-438	[&	488	^8	-488	^(538	a:	-538	a*	588	d<	-588	d,
439	[7	-439	['	489	^9	-489	^)	539	a;	-539	a+	589	d=	-589	d-
440	[8	-440	[(490	^:	-490	^*	540	a<	-540	a,	590	d>	-590	d.
441	[9	-441	[)	491	^;	-491	^+	541	a=	-541	a-	591	d?	-591	d/
442	[:	-442	[*	492	^<	-492	^,	542	a>	-542	a.	592	e0	-592	e ^{Sp}
443	[;	-443	[+	493	^=	-493	^-	543	a?	-543	a/	593	e1	-593	e!
444	[<	-444	[,	494	^>	-494	^.	544	b0	-544	b ^{Sp}	594	e2	-594	e"
445	[=	-445	[-	495	^?	-495	^ /	545	b1	-545	b!	595	e3	-595	e#
446	[>	-446	[.	496	_0	-496	_ ^{Sp}	546	b2	-546	b"	596	e4	-596	e\$
447	[?	-447	[/	497	_1	-497	_!	547	b3	-547	b#	597	e5	-597	e%
448	\0	-448	\ ^{Sp}	498	_2	-498	_"	548	b4	-548	b\$	598	e6	-598	e&
449	\1	-449	\!	499	_3	-499	_#	549	b5	-549	b%	599	e7	-599	e'

^a Positive integer
^b Negative integer

(continued)

ENCODED INTEGER PARAMETERS

Table D-1 (cont)

ENCODED INTEGER PARAMETERS

Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param
600	e8	-600	e (650	h:	-650	h *	700	k<	-700	k,	750	n>	-750	n.
601	e9	-601	e)	651	h;	-651	h +	701	k =	-701	k-	751	n?	-751	n /
602	e:	-602	e *	652	h<	-652	h,	702	k>	-702	k.	752	o0	-752	o ^{Sp}
603	e;	-603	e +	653	h =	-653	h-	703	k?	-703	k /	753	o1	-753	o!
604	e<	-604	e,	654	h>	-654	h.	704	10	-704	1 ^{Sp}	754	o2	-754	o"
605	e =	-605	e-	655	h?	-655	h /	705	11	-705	1!	755	o3	-755	o#
606	e>	-606	e.	656	i0	-656	i ^{Sp}	706	12	-706	1"	756	o4	-756	o\$
607	e?	-607	e /	657	i1	-657	i!	707	13	-707	1#	757	o5	-757	o%
608	f0	-608	f ^{Sp}	658	i2	-658	i"	708	14	-708	1\$	758	o6	-758	o&
609	f1	-609	f!	659	i3	-659	i#	709	15	-709	1%	759	o7	-759	o'
610	f2	-610	f"	660	i4	-660	i\$	710	16	-710	1&	760	o8	-760	o (
611	f3	-611	f#	661	i5	-661	i%	711	17	-711	1'	761	o9	-761	o)
612	f4	-612	f\$	662	i6	-662	i&	712	18	-712	1 (762	o:	-762	o *
613	f5	-613	f%	663	i7	-663	i /	713	19	-713	1)	763	o;	-763	o +
614	f6	-614	f&	664	i8	-664	i (714	1:	-714	1 *	764	o<	-764	o,
615	f7	-615	f'	665	i9	-665	i)	715	1;	-715	1 +	765	o =	-765	o-
616	f8	-616	f (666	i:	-666	i *	716	1<	-716	1,	766	o>	-766	o.
617	f9	-617	f)	667	i;	-667	i +	717	1 =	-717	1-	767	o?	-767	o /
618	f:	-618	f *	668	i<	-668	i,	718	1>	-718	1.	768	p0	-768	p ^{Sp}
619	f;	-619	f +	669	i =	-669	i-	719	1?	-719	1 /	769	p1	-769	p!
620	f<	-620	f,	670	i>	-670	i.	720	m0	-720	m ^{Sp}	770	p2	-770	p"
621	f =	-621	f-	671	i?	-671	i /	721	m1	-721	m!	771	p3	-771	p#
622	f>	-622	f.	672	j0	-672	j ^{Sp}	722	m2	-722	m"	772	p4	-772	p\$
623	f?	-623	f /	673	j1	-673	j!	723	m3	-723	m#	773	p5	-773	p%
624	g 0	-624	g ^{Sp}	674	j2	-674	j"	724	m4	-724	m\$	774	p6	-774	p&
625	g1	-625	g!	675	j3	-675	j#	725	m5	-725	m%	775	p7	-775	p'
626	g2	-626	g"	676	j4	-676	j\$	726	m6	-726	m&	776	p8	-776	p (
627	g3	-627	g#	677	j5	-677	j%	727	m7	-727	m'	777	p9	-777	p)
628	g4	-628	g\$	678	j6	-678	j&	728	m8	-728	m (778	p:	-778	p *
629	g5	-629	g%	679	j7	-679	j'	729	m9	-729	m)	779	p;	-779	p +
630	g6	-630	g&	680	j8	-680	j (730	m:	-730	m *	780	p<	-780	p,
631	g7	-631	g'	681	j9	-681	j)	731	m;	-731	m +	781	p =	-781	p-
632	g8	-632	g (682	j:	-682	j *	732	m<	-732	m,	782	p>	-782	p.
633	g9	-633	g)	683	j;	-683	j +	733	m =	-733	m-	783	p?	-783	p /
634	g:	-634	g *	684	j<	-684	j,	734	m>	-734	m.	784	q0	-784	q ^{Sp}
635	g;	-635	g +	685	j =	-685	j-	735	m?	-735	m /	785	q1	-785	q!
636	g<	-636	g,	686	j>	-686	j.	736	n0	-736	n ^{Sp}	786	q2	-786	q"
637	g =	-637	g-	687	j?	-687	j /	737	n1	-737	n!	787	q3	-787	q#
638	g>	-638	g.	688	k0	-688	k ^{Sp}	738	n2	-738	n"	788	q4	-788	q\$
639	g?	-639	g /	689	k1	-689	k!	739	n3	-739	n#	789	q5	-789	q%
640	h0	-640	h ^{Sp}	690	k2	-690	k"	740	n4	-740	n\$	790	q6	-790	q&
641	h1	-641	h!	691	k3	-691	k#	741	n5	-741	n%	791	q7	-791	q'
642	h2	-642	h"	692	k4	-692	k\$	742	n6	-742	n&	792	q8	-792	q (
643	h3	-643	h#	693	k5	-693	k%	743	n7	-743	n'	793	q9	-793	q)
644	h4	-644	h\$	694	k6	-694	k&	744	n8	-744	n (794	q:	-794	q *
645	h5	-645	h%	695	k7	-695	k'	745	n9	-745	n)	795	q;	-795	q +
646	h6	-646	h&	696	k8	-696	k (746	n:	-746	n *	796	q<	-796	q,
647	h7	-647	h'	697	k9	-697	k)	747	n;	-747	n +	797	q =	-797	q-
648	h8	-648	h (698	k:	-698	k *	748	n<	-748	n,	798	q>	-798	q.
649	h9	-649	h)	699	k;	-699	k +	749	n =	-749	n-	799	q?	-799	q /

^a Positive integer
^b Negative integer

(continued)

Table D-1 (cont)
ENCODED INTEGER PARAMETERS

Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param
800	r0	-800	r ^{Sp}	850	u2	-850	u"	900	x4	-900	x\$	950	{6	-950	{&
801	r1	-801	r!	851	u3	-851	u#	901	x5	-901	x%	951	{7	-951	{'
802	r2	-802	r"	852	u4	-852	u\$	902	x6	-902	x&	952	{8	-952	{(
803	r3	-803	r#	853	u5	-853	u%	903	x7	-903	x'	953	{9	-953	{)
804	r4	-804	r\$	854	u6	-854	u&	904	x8	-904	x(954	{:	-954	{*
805	r5	-805	r%	855	u7	-855	u'	905	x9	-905	x)	955	{;	-955	{+
806	r6	-806	r&	856	u8	-856	u(906	x:	-906	x*	956	{<	-956	{,
807	r7	-807	r'	857	u9	-857	u)	907	x;	-907	x+	957	{=	-957	{-
808	r8	-808	r(858	u:	-858	u*	908	x<	-908	x,	958	{>	-958	{.
809	r9	-809	r)	859	u;	-859	u+	909	x=	-909	x-	959	{?	-959	{/
810	r:	-810	r*	860	u<	-860	u,	910	x>	-910	x.	960	{0	-960	{ ^{Sp}
811	r;	-811	r+	861	u=	-861	u-	911	x?	-911	x/	961	{1	-961	{!
812	r<	-812	r,	862	u>	-862	u.	912	y0	-912	y ^{Sp}	962	{2	-962	{"
813	r=	-813	r-	863	u?	-863	u/	913	y1	-913	y!	963	{3	-963	{#
814	r>	-814	r.	864	v0	-864	v ^{Sp}	914	y2	-914	y"	964	{4	-964	{\$
815	r?	-815	r/	865	v1	-865	v!	915	y3	-915	y#	965	{5	-965	{%
816	s0	-816	s ^{Sp}	866	v2	-866	v"	916	y4	-916	y\$	966	{6	-966	{&
817	s1	-817	s!	867	v3	-867	v#	917	y5	-917	y%	967	{7	-967	{'
818	s2	-818	s"	868	v4	-868	v\$	918	y6	-918	y&	968	{8	-968	{(
819	s3	-819	s#	869	v5	-869	v%	919	y7	-919	y'	969	{9	-969	{)
820	s4	-820	s\$	870	v6	-870	v&	920	y8	-920	y(970	{:	-970	{*
821	s5	-821	s%	871	v7	-871	v'	921	y9	-921	y)	971	{;	-971	{+
822	s6	-822	s&	872	v8	-872	v(922	y:	-922	y*	972	{<	-972	{,
823	s7	-823	s'	873	v9	-873	v)	923	y;	-923	y+	973	{=	-973	{-
824	s8	-824	s(874	v:	-874	v*	924	y<	-924	y,	974	{>	-974	{.
825	s9	-825	s)	875	v;	-875	v+	925	y=	-925	y-	975	{?	-975	{/
826	s:	-826	s*	876	v<	-876	v,	926	y>	-926	y.	976	{0	-976	{ ^{Sp}
827	s;	-827	s+	877	v=	-877	v-	927	y?	-927	y/	977	{1	-977	{!
828	s<	-828	s,	878	v>	-878	v.	928	z0	-928	z ^{Sp}	978	{2	-978	{"
829	s=	-829	s-	879	v?	-879	v/	929	z1	-929	z!	979	{3	-979	{#
830	s>	-830	s.	880	w0	-880	w ^{Sp}	930	z2	-930	z"	980	{4	-980	{\$
831	s?	-831	s/	881	w1	-881	w!	931	z3	-931	z#	981	{5	-981	{%
832	t0	-832	t ^{Sp}	882	w2	-882	w"	932	z4	-932	z\$	982	{6	-982	{&
833	t1	-833	t!	883	w3	-883	w#	933	z5	-933	z%	983	{7	-983	{'
834	t2	-834	t"	884	w4	-884	w\$	934	z6	-934	z&	984	{8	-984	{(
835	t3	-835	t#	885	w5	-885	w%	935	z7	-935	z'	985	{9	-985	{)
836	t4	-836	t\$	886	w6	-886	w&	936	z8	-936	z(986	{:	-986	{*
837	t5	-837	t%	887	w7	-887	w'	937	z9	-937	z)	987	{;	-987	{+
838	t6	-838	t&	888	w8	-888	w(938	z:	-938	z*	988	{<	-988	{,
839	t7	-839	t'	889	w9	-889	w)	939	z;	-939	z+	989	{=	-989	{-
840	t8	-840	t(890	w:	-890	w*	940	z<	-940	z,	990	{>	-990	{.
841	t9	-841	t)	891	w;	-891	w+	941	z=	-941	z-	991	{?	-991	{/
842	t:	-842	t*	892	w<	-892	w,	942	z>	-942	z.	992	{~0	-992	{~ ^{Sp}
843	t;	-843	t+	893	w=	-893	w-	943	z?	-943	z/	993	{~1	-993	{~!
844	t<	-844	t,	894	w>	-894	w.	944	{0	-944	{ ^{Sp}	994	{~2	-994	{~"
845	t=	-845	t-	895	w?	-895	w/	945	{1	-945	{!	995	{~3	-995	{~#
846	t>	-846	t.	896	x0	-896	x ^{Sp}	946	{2	-946	{"	996	{~4	-996	{~\$
847	t?	-847	t/	897	x1	-897	x!	947	{3	-947	{#	997	{~5	-997	{~%
848	u0	-848	u ^{Sp}	898	x2	-898	x"	948	{4	-948	{\$	998	{~6	-998	{~&
849	u1	-849	u!	899	x3	-899	x#	949	{5	-949	{%	999	{~7	-999	{~/

^a Positive integer
^b Negative integer

(continued)

ENCODED INTEGER PARAMETERS

Table D-1 (cont)

ENCODED INTEGER PARAMETERS

Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param	Int ^a	Param	Int ^b	Param
1000	~8	-1000	~ (1050	AA:	-1050	AA *	1100	AD<	-1100	AD,	1150	AG>	-1150	AG.
1001	~9	-1001	~)	1051	AA;	-1051	AA +	1101	AD=	-1101	AD-	1151	AG?	-1151	AG /
1002	~:	-1002	~ *	1052	AA<	-1052	AA,	1102	AD>	-1102	AD.	1152	AH0	-1152	AH ^{Sp}
1003	~;	-1003	~ +	1053	AA =	-1053	AA-	1103	AD?	-1103	AD /	1153	AH1	-1153	AH!
1004	~<	-1004	~,	1054	AA>	-1054	AA.	1104	AE0	-1104	AE ^{Sp}	1154	AH2	-1154	AH"
1005	~ =	-1005	~-	1055	AA?	-1055	AA /	1105	AE1	-1105	AE!	1155	AH3	-1155	AH#
1006	~>	-1006	~.	1056	AB0	-1056	AB ^{Sp}	1106	AE2	-1106	AE"	1156	AH4	-1156	AH\$
1007	~?	-1007	~ /	1057	AB1	-1057	AB!	1107	AE3	-1107	AE#	1157	AH5	-1157	AH%
1008	D _r 0	-1008	D _r ^{Sp}	1058	AB2	-1058	AB"	1108	AE4	-1108	AE\$	1158	AH6	-1158	AH&
1009	D _r 1	-1009	D _r !	1059	AB3	-1059	AB#	1109	AE5	-1109	AE%	1159	AH7	-1159	AH'
1010	D _r 2	-1010	D _r "	1060	AB4	-1060	AB\$	1110	AE6	-1110	AE&	1160	AH8	-1160	AH(
1011	D _r 3	-1011	D _r #	1061	AB5	-1061	AB%	1111	AE7	-1111	AE'	1161	AH9	-1161	AH)
1012	D _r 4	-1012	D _r \$	1062	AB6	-1062	AB&	1112	AE8	-1112	AE(1162	AH:	-1162	AH*
1013	D _r 5	-1013	D _r %	1063	AB7	-1063	AB'	1113	AE9	-1113	AE)	1163	AH;	-1163	AH+
1014	D _r 6	-1014	D _r &	1064	AB8	-1064	AB(1114	AE:	-1114	AE*	1164	AH<	-1164	AH,
1015	D _r 7	-1015	D _r '	1065	AB9	-1065	AB)	1115	AE;	-1115	AE+	1165	AH=	-1165	AH-
1016	D _r 8	-1016	D _r (1066	AB:	-1066	AB*	1116	AE<	-1116	AE,	1166	AH>	-1166	AH.
1017	D _r 9	-1017	D _r (1067	AB;	-1067	AB+	1117	AE=	-1117	AE-	1167	AH?	-1167	AH/
1018	D _r :	-1018	D _r *	1068	AB<	-1068	AB,	1118	AE>	-1118	AE.	1168	AI0	-1168	AI ^{Sp}
1019	D _r ;	-1019	D _r +	1069	AB=	-1069	AB-	1119	AE?	-1119	AE/	1169	AI1	-1169	AI!
1020	D _r <	-1020	D _r ,	1070	AB>	-1070	AB.	1120	AF0	-1120	AF ^{Sp}	1170	AI2	-1170	AI"
1021	D _r =	-1021	D _r -	1071	AB?	-1071	AB/	1121	AF1	-1121	AF!	1171	AI3	-1171	AI#
1022	D _r >	-1022	D _r .	1072	AC0	-1072	AC ^{Sp}	1122	AF2	-1122	AF"	1172	AI4	-1172	AI\$
1023	D _r ?	-1023	D _r /	1073	AC1	-1073	AC!	1123	AF3	-1123	AF#	1173	AI5	-1173	AI%
1024	A@0	-1024	A@ ^{Sp}	1074	AC2	-1074	AC"	1124	AF4	-1124	AF\$	1174	AI6	-1174	AI&
1025	A@1	-1025	A@!	1075	AC3	-1075	AC#	1125	AF5	-1125	AF%	1175	AI7	-1175	AI'
1026	A@2	-1026	A@"	1076	AC4	-1076	AC\$	1126	AF6	-1126	AF&	1176	AI8	-1176	AI(
1027	A@3	-1027	A@#	1077	AC5	-1077	AC%	1127	AF7	-1127	AF'	1177	AI9	-1177	AI)
1028	A@4	-1028	A@\$	1078	AC6	-1078	AC&	1128	AF8	-1128	AF(1178	AI:	-1178	AI*
1029	A@5	-1029	A@%	1079	AC7	-1079	AC'	1129	AF9	-1129	AF)	1179	AI;	-1179	AI+
1030	A@6	-1030	A@&	1080	AC8	-1080	AC(1130	AF:	-1130	AF*	1180	AI<	-1180	AI,
1031	A@7	-1031	A@'	1081	AC9	-1081	AC)	1131	AF;	-1131	AF+	1181	AI=	-1181	AI-
1032	A@8	-1032	A@(1082	AC:	-1082	AC*	1132	AF<	-1132	AF,	1182	AI>	-1182	AI.
1033	A@9	-1033	A@)	1083	AC;	-1083	AC+	1133	AF=	-1133	AF-	1183	AI?	-1183	AI/
1034	A@:	-1034	A@*	1084	AC<	-1084	AC,	1134	AF>	-1134	AF.	1184	AJ0	-1184	AJ ^{Sp}
1035	A@;	-1035	A@+	1085	AC=	-1085	AC-	1135	AF?	-1135	AF/	1185	AJ1	-1185	AJ!
1036	A@<	-1036	A@,	1086	AC>	-1086	AC.	1136	AG0	-1136	AG ^{Sp}	1186	AJ2	-1186	AJ"
1037	A@=	-1037	A@-	1087	AC?	-1087	AC/	1137	AG1	-1137	AG!	1187	AJ3	-1187	AJ#
1038	A@>	-1038	A@.	1088	AD0	-1088	AD ^{Sp}	1138	AG2	-1138	AG"	1188	AJ4	-1188	AJ\$
1039	A@?	-1039	A@/	1089	AD1	-1089	AD!	1139	AG3	-1139	AG#	1189	AJ5	-1189	AJ%
1040	AA0	-1040	AA ^{Sp}	1090	AD2	-1090	AD"	1140	AG4	-1140	AG\$	1190	AJ6	-1190	AJ&
1041	AA1	-1041	AA!	1091	AD3	-1091	AD#	1141	AG5	-1141	AG%	1191	AJ7	-1191	AJ'
1042	AA2	-1042	AA"	1092	AD4	-1092	AD\$	1142	AG6	-1142	AG&	1192	AJ8	-1192	AJ(
1043	AA3	-1043	AA#	1093	AD5	-1093	AD%	1143	AG7	-1143	AG'	1193	AJ9	-1193	AJ)
1044	AA4	-1044	AA\$	1094	AD6	-1094	AD&	1144	AG8	-1144	AG(1194	AJ:	-1194	AJ*
1045	AA5	-1045	AA%	1095	AD7	-1095	AD'	1145	AG9	-1145	AG)	1195	AJ;	-1195	AJ+
1046	AA6	-1046	AA&	1096	AD8	-1096	AD(1146	AG:	-1146	AG*	1196	AJ<	-1196	AJ,
1047	AA7	-1047	AA'	1097	AD9	-1097	AD)	1147	AG;	-1147	AG+	1197	AJ=	-1197	AJ-
1048	AA8	-1048	AA(1098	AD:	-1098	AD*	1148	AG<	-1148	AG,	1198	AJ>	-1198	AJ.
1049	AA9	-1049	AA)	1099	AD;	-1099	AD+	1149	AG=	-1149	AG-	1199	AJ?	-1199	AJ/

^a Positive integer
^b Negative integer

(continued)

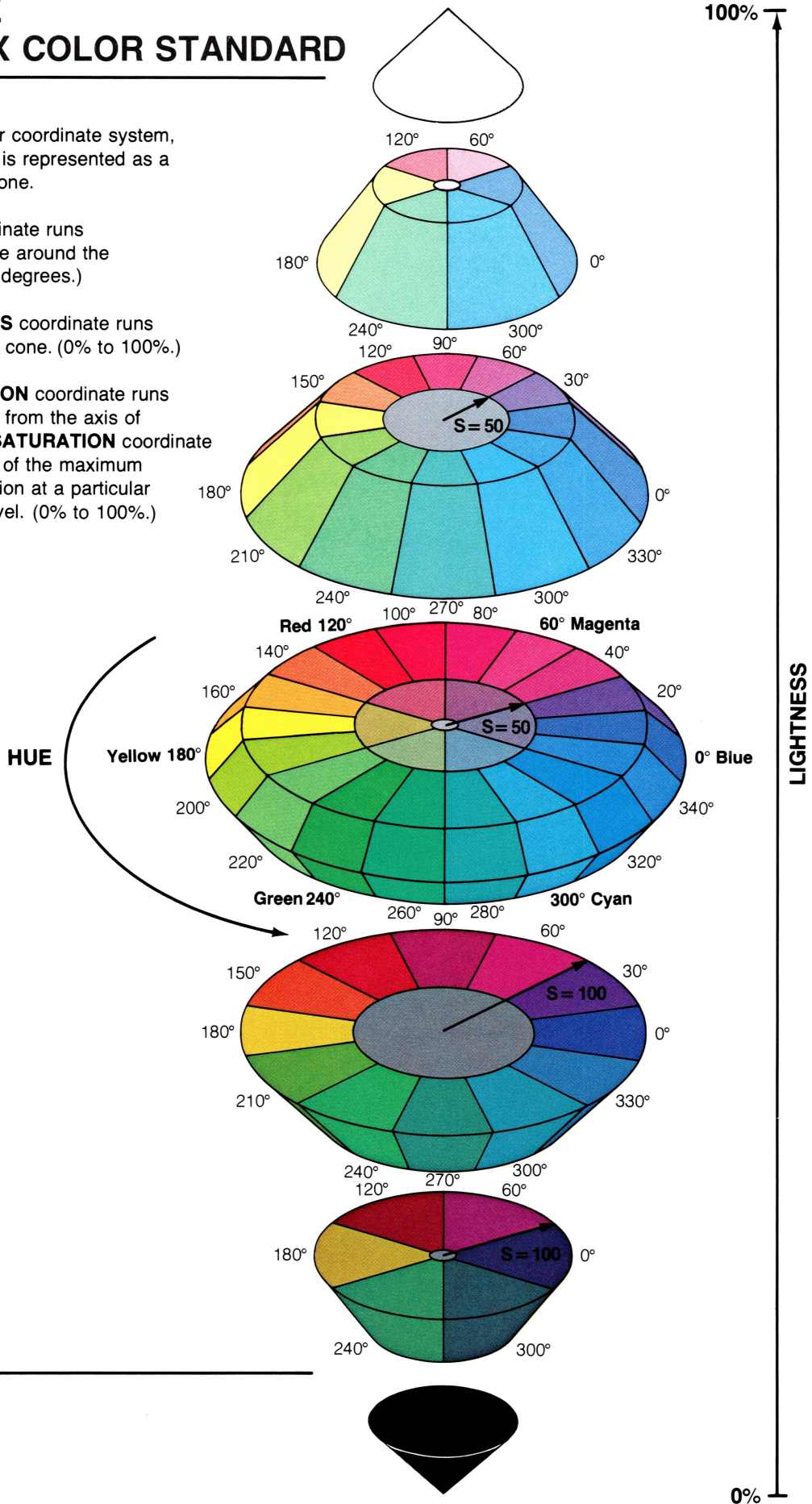
Appendix E TEKTRONIX COLOR STANDARD

In the **HLS** color coordinate system, the color space is represented as a double-ended cone.

The **HUE** coordinate runs counterclockwise around the cone. (0 to 360 degrees.)

The **LIGHTNESS** coordinate runs vertically up the cone. (0% to 100%.)

The **SATURATION** coordinate runs radially outward from the axis of the cone. The **SATURATION** coordinate is a percentage of the maximum possible saturation at a particular **LIGHTNESS** level. (0% to 100%.)



(continued)

TEKTRONIX COLOR STANDARD

Overview:

The world of color is filled with ambiguous terminology, i.e. intensity, purity, value, etc. Many color users feel that "color theory" is a prerequisite to operating color systems; T.V., Videotaping, Photography, Computer Graphics.

In order to end this confusion, Tektronix has developed a color language and function based on human engineering, rather than machine engineering. Below is a description of this system, which will provide a clear and concise means for understanding how color is defined and how our syntax was derived.

Color Concepts:

Color selection is specified by hue, lightness and saturation which is the HLS method. The definitions are as follows:

Hue: The characteristic associated with a color name such as red, yellow, green, blue, etc. Hue is a gradation of color advanced by degrees, thus represented as an angle from 0 to 360.

Lightness: The characteristic that allows the color to be ranked on a scale from dark to light. Lightness is expressed as a parameter ranging from 0 to 100% with black being 0 (bottom of cone) and white being 100% (top of cone).

Saturation: The characteristic which describes the extent to which a color differs from a gray of the same lightness. Saturation is expressed as percentage, ranging from 0% (maximum white content at that lightness level) to 100% (full saturated).

Geometrically, colors can be described in terms of a double cone.

Variations in lightness are represented along the axis, with white at the apex of the cone and black at the opposite apex. Variations in saturation are represented by radial distances from the lightness axis, in constant lightness planes. Hue is represented as an angular quantity from a known reference point.

Copyright © 1982 by Tektronix, Inc., Beaverton, Oregon. Printed in the United States of America. All rights reserved. Contents of this publication may not be reproduced in any form without permission of Tektronix, Inc. U.S.A. and foreign TEKTRONIX products covered by U.S. and foreign patents and/or patents pending.

TEKTRONIX is a registered trademark for Tektronix, Inc.

Appendix F PREDEFINED FILL PATTERNS

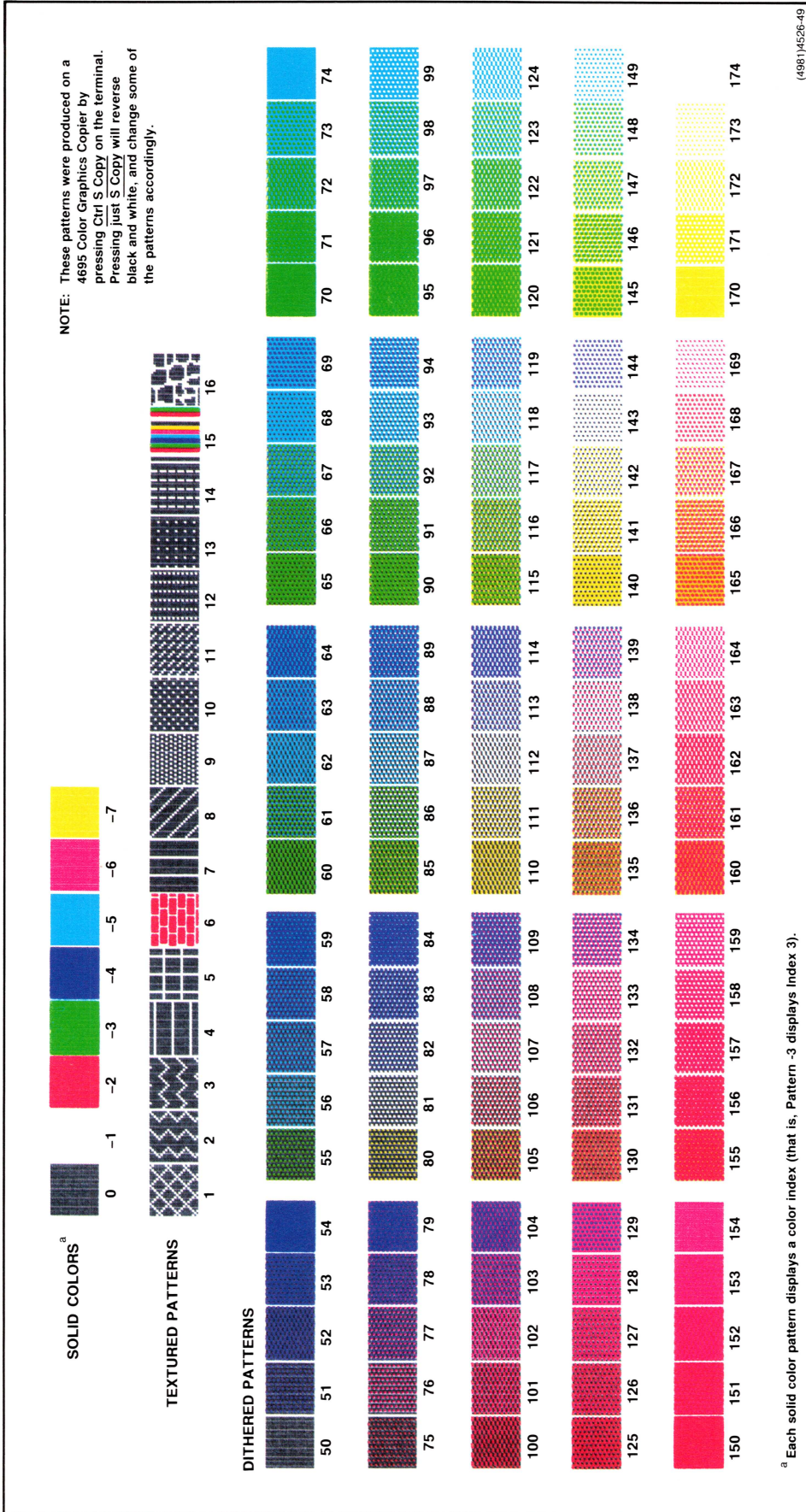


Figure F-1. Predefined Fill Patterns.

(continued)

FILL PATTERNS

NOTES ON FILL PATTERNS

Your terminal uses predefined patterns to fill the *panels* (polygons) that you create. A sample of each of these fill patterns, along with its pattern number, is shown in Figure F-1.

To use one of the patterns, choose the pattern number you want from Figure F-1 and issue the SELECT FILL PATTERN command using that number. The terminal will fill the next panel you draw with the fill pattern you have selected. (You create panels by issuing a BEGIN PANEL BOUNDARY command and then issuing MOVE and DRAW commands to draw the panel boundary. The terminal doesn't fill in the panel until you issue an END PANEL command.)

The terminal uses colors from the graphics area color map to create fill patterns. Figure F-1 uses the factory default color values, which are listed in the command description for SET SURFACE COLOR MAP. If the color map is changed from its factory default settings, the appearance of some or all of these patterns will change accordingly. You can change the color map by issuing the SET SURFACE COLOR MAP command or by using the Interactive Color Interface.



Appendix G

THE 4104 TERMINAL

The 4104 Computer Display Terminal offers the same set of commands as the 4105, and it supports all the same peripheral devices. It differs from the 4105 in these ways:

- The 4104 has two bit planes instead of three; thus, there are only four graphics color indices available and four dialog area color indices.
- The 4104 does not have the Interactive Color Interface; thus, the user can only control colors from the keyboard by issuing Setup commands.

Applications written for the 4105 will run as written on the 4104, but some commands will not execute and will generate error messages. This appendix describes how these differences affect applications written for the 4105, and describes how you can ensure that applications you write will run on both the 4105 and 4104.

BIT PLANES AND COLOR INDICES

Since the 4104 has two bit planes, there are four graphics color indices and four dialog area color indices — instead of the eight of each available on the 4105. The default values assigned to these color indices are:

- 0 Black
- 1 White
- 2 Red
- 3 Green

These are the same as the default values for Indices 0 through 3 on the 4105. (The 4104 Terminal can still display the same range of color mixtures, and you can use the SET SURFACE COLOR MAP and SET DIALOG AREA COLOR MAP commands to assign different color mixtures to any of the four available indices.)

HOW THE 4104 HANDLES OUT-OF-RANGE INDEX NUMBERS

With most commands, the terminal defaults to the highest valid index when you specify an index number higher than 3. The following commands work in this way:

- SET ALPHA CURSOR INDICES
- SET DIALOG AREA INDEX
- SET LINE INDEX
- SET TEXT INDEX
- SET VIEW ATTRIBUTES
- SET SURFACE COLOR MAP

Because the 4104 Terminal maps all higher indices to Index 3, an application written for a 4105 will run as written on a 4104. However, since Indices 4 through 7 all map to Index 3 (green, by default) screen images that were designed for display on the 4105 may not be as effective on the 4104. Thus, if you are writing an application to run on both 4104 and 4105 Terminals, you should ensure that the displays are still effective when using only Indices 0 through 3.

Three other commands handle out-of-range index numbers differently — in some cases, they generate error messages. The error messages are displayed on the terminal screen, unless you set the terminal's error threshold to 3 (see the SET ERROR THRESHOLD command description).

Here are these commands and how they work with the 4104:

SET DIALOG AREA COLOR MAP. With this 4100-style command, values 4 through 7 are invalid for the index parameter, and issuing them generates Error TF11.

THE 4104 TERMINAL

SELECT FILL PATTERN. This 4100-style command lets you choose several fill patterns that use graphics color indices. The terminal maps the invalid indices to valid ones where possible:

- Fill Patterns -4 through -7 (which specify Color Indices 4 through 7) wrap in a modulo function, so:
 - -4 maps to Index 0
 - -5 maps to Index 1
 - -6 maps to Index 2
 - -7 maps to Index 3
- The “rainbow” fill pattern (Fill Pattern 15) has four colors instead of eight.
- Fill patterns 50 through 174 (the dithered patterns) are not available, and requesting them generates Error MP10.

SGR (SELECT GRAPHICS RENDITION). With this ANSI command, how the terminal treats requests for Indices 4 through 7 depends on whether you use digit-only parameters or prefixed parameters:

- The prefixed parameter values <4 through <7 , >4 through >7 , and $=4$ through $=7$ are valid and all select Index 3.
- Digit-only parameter values 33 through 36 and 43 through 46 are invalid, and issuing them generates Error [m11].

REPORTS

The only report that works differently in the 4104 is the Terminal Settings Report. When you request a report of terminal type (special inquiry code $?T$), the Terminal Settings Report indicates that the terminal is a 4104 — if your application tests for terminal type, you’ll need to change your test to make it accept *4104* as a valid terminal type.

GLOSSARY

4010 GIN Report

A report the terminal sends to the host during *GIN* (graphics input) when the user presses any keyboard key that sends an ASCII character. The report contains the GIN cursor location (as a *4010 XY-report parameter*) and the identity of the key the user pressed.

4010 Status Report

A report the terminal sends the host in response to a REPORT 4010 STATUS command. The report contains the alpha cursor position and the status of the copier, or the GIN cursor position if 4010 GIN is enabled. Cursor position is reported as a *4010 XY-report parameter*.

4010 XY-report parameter

A type of *parameter* that the terminal uses when it makes 4010 GIN Reports and 4010 Status Reports to the host. The 4010 XY-report parameter is a four character string of ASCII characters that uses a 1024 by 1024 coordinate space to represent the screen coordinates of the GIN cursor. Compare to *XY-coordinate report parameter*.

4100-style commands

One of the *command sets* available on your terminal. This set includes the graphics commands, the communications commands, and the commands that set up dialog display and keyboard characteristics. The 4100-style commands are defined by Tektronix; a number of other manufacturer's graphics terminals support a subset of the 4100-style commands.

ADE

ASCII decimal equivalent, which is the numeric value used to represent an alphanumeric character or *control character*. The code charts in Appendix A show each character's ADE values (the ADE is shown to the lower-right of each character).

alpha cursor

An underscore displayed on the terminal's screen during screen editing, used to indicate where characters entered from the keyboard will be written.

Alpha mode

One of the terminal's three *implicit command modes*. In Alpha mode, the terminal interprets all incoming characters as alphanumerics and writes them to the *current location* in the *dialog buffer* (if the dialog area is disabled, the terminal writes the alphanumerics to the graphics area).

alphanumeric memory

The area of the terminal's program memory that is used for storing and manipulating alphanumeric data.

alphanumerics

One of two styles of text. Alphanumerics is made up of 5 by 7 pixel dot matrix characters and is used primarily in the dialog area, as opposed to *graphtext*, which is used in the graphics area. See also *character set*.

ALU

Abbreviation for arithmetic logic unit.

ANSI

The American National Standards Institute. This is the USA member body of the International Organization for Standardization (ISO). ANSI Standard X3.64 specifies commands for use with ASCII data, including commands to format and edit text on a terminal's display.

ANSI mode

One of the terminal's *host command modes*. ANSI mode allows the terminal to interpret ANSI Standard X3.64 screen-editing commands and commands that control dialog display and keyboard characteristics.

answerback string

A password-like string of characters that the terminal transmits to the host upon request. This security feature allows the host to control the terminal's access to programs and data.

array count

In *integer arrays*, an encoded integer that specifies how many integer parameters follow it in the array.

ASCII

The American Standard Code for Information Interchange. ASCII is a scheme of data representation, established by *ANSI*, in which characters and control codes are represented by seven-bit patterns.

autoprint

See *data logging*.

autorepeat

The capability to transmit a character repeatedly by holding a key down. With autorepeat on, characters or strings are repeated when keys are held down; with autorepeat off, characters and strings transmit only once, even if the key that transmits them is held down.

Italicized words within a definition are terms defined elsewhere in this glossary.

GLOSSARY
AUTOWRAP

autowrap

The capability to wrap (continue) text from the end of one line to the beginning of the next line without the need for explicit Carriage Return (CR) and Line Feed (LF) characters.

baud rate

A measure of the speed of data communication; in RS-232 communications, a speed of 300 baud corresponds to 300 bits per second.

bit plane

A layer of *graphics memory* that includes one bit for each addressable location on the terminal screen. The 4105 terminal has three bit planes.

break signal

The signal that interrupts communications between the terminal and a host. The break signal is sent when the terminal's Break key is pressed.

buffering

The process of storing transmitted data in memory so that it can be processed at a later time.

Bypass mode

One of the terminal's *communications modes*. Bypass mode allows the terminal to ignore data sent from the host — used mainly to avoid *echoing* reports sent to the host.

character cell

The rectangular area (6 by 12 pixels) surrounding each alphanumerical character.

character flagging

A method of *software flagging* that uses two user-defined characters, one to indicate the beginning of data transmission and another to indicate the ending of data transmission. See *start character* and *stop character*.

character parameter

A type of parameter that the terminal uses when it reports character data to the host. A *character report parameter* is an ASCII character whose ADE is in the range 0 through 127.

character set

A predefined set of alphanumeric symbols. The terminal contains a character set for each available language-dependent keyboard, as well as a set of supplementary symbols and a set of rulings characters. See also *alphanumerical*, *graphtext*, and *font*.

clipping

The truncating of an image when part of it is displayed outside a *window*.

color index

A number, from 0 through 7, that specifies a location in the terminal's *color map*, which stores *color mixtures*. You can change the color mixture assigned to an index by using 4100-style commands or the *Interactive Color Interface*.

color map

A table in the terminal's program memory that stores a *color mixture* definition for each of the terminal's color indices. The terminal uses the color map to translate a *color index* number into a color on the display.

color mixture

A color uniquely identified by a *color index* and defined in the *HLS color coordinate system* by a combination of hue, lightness, and saturation.

command set

A group of commands that use the same syntax. The 4105 terminal recognizes three command sets: ANSI, VT52, and 4100-style. See also *host syntax* and *Setup syntax*.

communications modes

The three modes that control how the terminal communicates with the host. The communications modes are *Bypass mode*, *Local mode*, and *Prompt mode*.

control character

A character that initiates, modifies, or stops an operation. Control characters are not normally displayed when transmitted.

COPIER port

A terminal port with associated firmware that lets you connect Tektronix color graphics copiers and monochrome graphics printers to the terminal.

current line

In screen editing, the display line containing the *alpha cursor*.

current position

In screen editing, the row-and-column position of the *alpha cursor*.

cursor

A symbol displayed on the terminal's screen to indicate where characters or other input entered from the keyboard will be placed. This terminal has an *alpha cursor* for screen editing and a *GIN cursor* for graphics input (GIN).

Italicized words within a definition are terms defined elsewhere in this glossary.

data logging

The process of writing data simultaneously to the terminal's dialog area and a copier or printer.

DC1/DC3 flagging

A method of *software flagging* that uses the D_1 (ADE 17) character and the D_3 (ADE 19) character to indicate the beginning and ending of data transmission. DC1/DC3 flagging is used to control host communications. See also *start character* and *stop character*.

delimited string parameter

In Setup commands, a parameter type that consists of a pair of delimiter characters enclosing a string of one or more characters.

delimiter

A character that marks the beginning and ending of a string of characters; the delimiter must not be contained in the string it delimits.

dialog

Communications (prompts, user entries, error messages) exchanged between a host program and a user.

dialog area

A variable part of the screen that is superimposed over the *graphics area* to display dialog between a host and a user. The dialog area displays text when using text editing programs.

dialog area buffer

The area of the terminal's program memory that is used to store information to be displayed in the *dialog area*. Often called simple *dialog buffer*.

dither pattern

A type of *fill pattern*. A dither pattern combines two or more colors in a pattern to create the illusion of another color.

DTR/CTS flagging

A method of *hardware flagging* used in host communications. DTR/CTS flagging uses the RS-232 Data Terminal Ready (DTR) and Clear to Send (CTS) signals to control the flow of data.

echo

The retransmission of a character entered at the terminal's keyboard, used to display characters on the terminal's screen. Echo can be provided by the terminal or by the host. See *local echo* and *remote echo*.

edit margins

The rows of the terminal screen that have been designated as the upper and lower boundaries of the *scrolling region*. Defined by the ANSI command TEKSTBM (SET TOP AND BOTTOM MARGINS).

EDIT mode

One of the terminal's *host command modes*. EDIT mode is an implementation of ANSI mode in which the terminal characteristics are automatically set to emulate a VT100 terminal.

erase index

A *color index* used for erasing images in the graphics area. The erase index is always Index 0. When an image is traced over in Index 0, the image is erased from the screen.

Error Report

A *report* the terminal sends the host in response to a REPORT ERRORS command. The Error Report contains the error codes for the eight most recently detected errors, along with information about the severity of the errors, and a count of how many times each error was detected.

escape sequence

A series of characters beginning with an E_c character; the E_c is usually followed by an *opcode*. Most 4100-style commands require an escape sequence as part of their host syntax.

fill patterns

The terminal's predefined patterns and solid colors used for filling *panels*. See Appendix F for illustrations of available fill patterns.

filler characters

Characters that some host computers insert in a data stream to adjust the pace of data communications. Filler characters are usually N_V (ADE 0) or D_T (ADE 127).

fixed region

In screen editing applications, an area of the dialog area buffer that cannot be scrolled. See also *scrolling region*.

flagging

A method of data transmission in which a device or program activates signals or inserts characters in a stream of data to control the flow of data. See also *hardware flagging* and *software flagging*.

Italicized words within a definition are terms defined elsewhere in this glossary.

GLOSSARY
FONT

font

A set of alphanumeric or graphics symbols.

function keys

The terminal keys that invoke specialized functions with a single keystroke. The function invoked by a specific function key is determined by the software or application program in use.

G0 character set

The character set invoked by issuing the $\$i$ (Shift In) ASCII control character. The ANSI command SCS (SELECT CHARACTER SET) assigns the G0 character set.

G1 character set

The character set invoked by issuing the $\$o$ (Shift Out) ASCII control character. The ANSI command SCS (SELECT CHARACTER SET) assigns the G1 character set.

GIN

The terminal's unique method of processing graphics input. When GIN is enabled, users *input* a point and the terminal converts the location of that point into a digital form that can be transmitted to and interpreted by a host computer. The 4105 terminal emulates 4010 GIN.

GIN cursor

A crosshair symbol used during GIN to indicate locations on the screen. The user presses the Joydisk to move the GIN cursor.

graphic rendition

Characteristics for *alphatext*; choices include character color, background color, bold, underscore, slow blink, and reverse video. Set with the ANSI SGR (SELECT GRAPHIC RENDITION) command.

graphics area

The image of graphics memory displayed on the terminal screen. The dialog area is displayed in front of the graphics area and may obscure part of it. Compare to *dialog area*.

graphics memory

The part of terminal memory that stores images displayed on the terminal's screen. Graphics memory includes both *on-screen* and *off-screen memory*.

graphics position

The position in *terminal space* where the last graphics operation left off. At power-up, the graphics position is at home (terminal space coordinates 0,3071).

graphics primitives

The fundamental elements of a graphics image: markers, panels, vectors, *graphtext*, and *alphatext* displayed in the graphics area.

graphtext

One of two types of text. Graphtext is displayed in the graphics area and can be resized, rotated, and written in different directions. Compare to *alphatext*.

handshaking

A formalized sequence of operations for pacing the flow of data between the terminal and a host or peripheral. See also *flagging* and *Prompt mode*.

hard copy

A physical copy (typically printed on paper) of alphanumeric or graphics data.

hardware flagging

A *flagging* scheme in which a device signals its readiness to transmit or receive data by the presence or absence of an electrical signal under hardware control. Hardware flagging on this terminal is DTR/CTS flagging, used for host communications. See also *software flagging*.

HLS color coordinate system

A method of representing a color mixture by specifying *hue*, *lightness*, and *saturation*.

home

A predefined position on the terminal screen (usually the upper-left corner) to which the cursor can be returned with a single command. This terminal's home position for graphics is at terminal space coordinates 0,3071. For screen editing applications, home position is Row 1, Column 1 of the dialog area buffer (or Row 1, Column 1 of the scrolling region if one is defined and Origin mode is Relative).

host

A computer device that controls the exchange of data between itself and another device.

host command modes

The terminal's four modes that determine which command set the terminal will accept. The host command modes are *ANSI mode*, *Edit mode*, *TEK mode*, and *VT52 mode*.

host port

The connector and the associated firmware that permit the terminal to communicate with a host computer.

Italicized words within a definition are terms defined elsewhere in this glossary.

host syntax

The conventions that a program running on a host computer must use to send commands to the terminal. Compare to *Setup syntax*.

hue

The first coordinate of the HLS color system. Hue specifies color as an angle between 0° and 360°; for example, 120° specifies red.

implicit command modes

The terminal's three modes — *Alpha mode*, *Marker mode*, and *Vector mode* — that let you omit certain command's escape sequences and issue parameter values only. See *Alpha mode*, *Marker mode*, and *Vector mode*.

Index 0

A *color index* reserved for special purposes. In the dialog area Index 0 is used to make the background transparent so the graphics area shows through, and in the graphics area, Index 0 is used as the *erase index*.

input, GIN

In 4010 GIN, the user action that sends a 4010 GIN report to the host. The user presses the Joydisk to position the GIN cursor and then presses any keyboard key that sends an ASCII character to input the GIN cursor location.

input queue

The part of the terminal's program memory that is used to store data received from other devices. See also *buffering* and *output queue*.

integer array

A sequence of integer parameters that consists of an *array count* followed by one or more integer parameters. You use integer arrays in issuing some TEK mode commands from the host.

integer parameter

A type of *parameter* that a host program must use to encode integer values when issuing TEK mode commands from the host (does not use the same encoding scheme as the *integer report parameter*).

integer report parameter

A type of *parameter* that the terminal uses to encode integer report values when it sends a *report* to the host (does not use the same encoding scheme as the *integer parameter*).

Interactive Color Interface

Firmware in the 4105 terminal that lets you change the colors of the display from the terminal keyboard. See also *color map*.

interface

Software, firmware, or hardware designed to allow different computer software, firmware, or hardware systems to exchange data and control information.

Joydisk

An octagonal control on the terminal keyboard used to move the GIN cursor in the graphics area or to scroll text in the dialog area.

key combination

The combination of the Ctrl key and/or the Shift key with another keyboard key (holding down the Ctrl key, the Shift key, or the Ctrl-Shift combination while pressing another keyboard key causes the key to transmit different values than it would when pressed alone).

key macro

A *macro* that can be invoked from the keyboard by pressing a single key or a key combination.

key specifier

A type of parameter used in Setup to specify a key by just pressing it. Key specifiers are used for defining macros.

lightness

The second coordinate in the HLS color coordinate system. Lightness is the proportion of white in a *color mixture*, specified as a percentage between 0% (pure black) and 100% (pure white).

local echo

The terminal's retransmission of characters entered at its keyboard, used to display characters on the terminal's screen. Compare to *remote echo*.

Local mode

A communications mode that lets you enter commands from the keyboard using host syntax rather than Setup syntax. In Local mode, the terminal does not transmit data to the host; any data sent from the host is stored in the terminal's input queue.

macro

A sequence of commands or data that has been predefined so that it can be invoked by a single command or keystroke.

marker

A predefined symbol used to identify important points in a drawing, such as data points on a graph or cities on a map.

Italicized words within a definition are terms defined elsewhere in this glossary.

MARKER MODE

Marker mode

One of the terminal's three *implicit command modes*. In Marker mode, the terminal interprets groups of incoming characters as xy-coordinates and draws markers at the indicated locations.

memory

A part of a computer, terminal, or other device that stores data. See also, *graphics memory*, *nonvolatile memory*, and *program memory*.

mode

A terminal setting that determines how the terminal treats commands. See *communications modes*, *host command modes*, and *implicit command modes*.

nonvolatile macro

A macro that is stored in the terminal's *nonvolatile memory*, which means that the terminal retains it from session to session, even after being turned off or reset. Compare to *volatile macro*.

nonvolatile memory

The part of the terminal's program memory that can save command settings and macro definitions even while the terminal is turned off. In the command descriptions in this manual, the phrase *Can be saved in nonvolatile memory* identifies commands whose settings can be saved.

numeric keypad

A group of keys on the terminal keyboard that are laid out like a standard ten-key adding machine. The numeric keypad can be used for numeric data entry or to send predefined codes, which you can use to invoke application functions.

off-screen memory

An area of *graphics memory* that is not displayable.

on-screen memory

The displayable portion of *graphics memory*.

opcode

The two characters that follow the E_c character in a 4100-style command's escape sequence. Each command's opcode is unique.

Origin mode

A terminal mode that determines how the cursor is addressed in the dialog area buffer. Origin mode determines whether cursor addresses are based on the first position in the *dialog area buffer* or the first position in the *scrolling region*.

output queue

The part of the terminal's program memory that is used to store data to be transmitted to other devices. See also *buffering* and *input queue*.

overstrike

The terminal's ability to display a character superimposed on another character without erasing the original character. In screen editing, this capability is limited to the Space and Underscore characters.

panel

A polygonal boundary defined by a series of MOVE and DRAW commands. A panel may be a simple closed outline, or it may be filled with a solid color or a pattern. See also *fill patterns*.

parameter

The variable part of a command. Parameter values represent data or a choice between options in many commands. You must encode parameter values used in TEK mode commands issued from the host; you use integers or simple keywords as parameter values in Setup commands issued from the keyboard.

parameter type

The type of *parameter* that a command requires. The parameter type identifies the data format that must be used to issue the parameter in a specific syntax. See also, *4010 XY-report parameter*, *character parameter*, *integer array parameter*, *integer parameter*, *integer report parameter*, *key specifier*, *string parameter*, *XY-coordinate parameter* and *XY-coordinate report parameter*.

parity bit

A bit transmitted in each byte of data to indicate whether the sum of the bits in the byte is odd or even. The parity bit is set by the transmitting device and enables the receiving device to verify the transmitted data.

peripheral device

A device used to provide input data or process output data for a computer or terminal.

pixel

The smallest unit of a terminal's screen display that can be addressed or assigned attributes.

pixel beam position

The position in the pixel viewport at which the RASTER WRITE and RUNLENGTH WRITE operations will begin.

Italicized words within a definition are terms defined elsewhere in this glossary.

pixel viewport

A user-defined rectangular area of graphics memory, addressed in pixel units. The pixel viewport defines the limits of the pixel beam position (that is, pixel beam position 0,0 is the lower left corner of the viewport, and issuing pixel beam coordinates outside the pixel viewport causes the pixel beam position to move to the point in the pixel viewport closest to the specified location).

port

A physical connection (such as a multipin connector) along with the associated firmware that permits one computing device to exchange data and commands with another computing device or peripheral device.

program memory

The terminal's general-purpose, random-access memory. Host programs can use program memory to define panels, to define and store macro definitions, or to allocate more space to the input queue or dialog area buffer. Program memory is comprised of *volatile memory* and *nonvolatile memory*.

Prompt mode

A communications mode that uses a *prompt string* and a *transmit delay* to control transmission of single lines of data from the terminal to the host.

prompt string

A string of characters sent by the host during *Prompt mode* to tell the terminal that the host is ready to send data. The terminal must be configured to recognize the prompt string sent by the host computer.

random-access memory (RAM)

Memory that a program can use for temporary storage of command settings and graphics data. An application program can write data to random access memory as well as read data from it. Compare to *read-only memory*.

read-only memory (ROM)

Memory that the terminal uses for permanent storage of operating firmware. An application program can read data from read-only memory but can not alter the data there. Compare to *random access memory*.

remote echo

The host's retransmission of characters entered at the terminal's keyboard, used to display characters on the terminal's screen. Compare to *local echo*.

report

Data sent from the terminal to the host, in response to a host command. Reports contain information about terminal settings or status.

Italicized words within a definition are terms defined elsewhere in this glossary.

routine

Programming code used by one or more programs to perform a specific common function. Section 6 includes routines that you can use to encode parameters and to decode reports.

RS-232-C

A standard communications *interface* defined by the Electronic Industries Association (EIA) to control data communications using standard voltages, signal lines, and device interactions. (Sometimes abbreviated in this manual as RS-232.)

saturation

The third coordinate in the HLS color coordinate system. Saturation is the intensity of hue in a color mixture and ranges from 0% (neutral — white, gray, or black) to 100% (pure color).

screen editor

A text entry and editing program that allows you to view and edit a computer file as a whole rather than just line-by-line.

scrolling

Vertical or horizontal movement of text on the terminal screen. Scrolling permits easy reading of blocks of text larger than the screen can accommodate.

scrolling region

An area of the dialog area buffer used in screen editing. When the scrolling region is full, new lines of data are written at the end of the scrolling region, and the lines of data that were at the top of the scrolling region disappear. Compare to *fixed region*.

Self Test

A program stored in the terminal's firmware that tests terminal functions and displays any errors it finds. There are three types of self test: power-up, extended, and adjustment. See your Operators Manual for details.

Setup

A method of entering certain commands locally from the keyboard. In Setup, the terminal accepts keyboard entry of commands using *Setup syntax*. Setup is selected from the terminal keyboard by pressing the Setup key.

Setup syntax

The form of a command that is entered at the terminal keyboard. Some ANSI and VT52 commands and most 4100-style commands have Setup syntax; you can issue any such command from the keyboard without regard for the terminal's host command mode. Compare to *host syntax*.

SNOOPY MODE

Snoopy mode

A mode in which the terminal displays (rather than executes) ASCII control characters transmitted from the host or entered at the keyboard.

software flagging

A *flagging* scheme in which a device's readiness to transmit or receive data is signaled by specific characters under software control. Software flagging for this terminal is *DC1/DC3 flagging*, used for host communications. Compare to *hardware flagging*.

special inquiry code

A code included in the REPORT TERMINAL SETTING command to cause the terminal to report the status of program memory, the terminal model number, or the firmware version number.

stop bit

One or two bits transmitted in each byte of data to indicate the end of the byte.

string parameter

A type of parameter that the terminal uses when it reports arrays of character data to the host. The string report parameter consists of ASCII characters preceded by an integer report that contains the number of characters in the string. If there are no characters in the string (an empty string), the count is 0.

TEK mode

One of the terminal's *host command modes*. TEK mode allows the terminal to interpret 4100-style commands.

Terminal Settings Report

A *report* the terminal sends the host in response to a REPORT TERMINAL SETTING command. The Terminal Settings Report contains the current settings or status for a specific command, or special information about program memory availability, terminal model, or firmware level.

terminal space

An imaginary plane on which you draw graphics images. Terminal space is bounded by the addressing limits of the graphics commands — 4096 by 4096 *terminal space units*.

terminal space coordinates

A pair of integers that give the XY-coordinates (in *terminal space units*) of a location in *terminal space*. You must encode terminal space coordinates as *XY-coordinate parameters* when you issue TEK mode commands from the host.

terminal space unit (TSU)

The unit of measure used for defining dimensions in terminal space.

transmit delay

A specified delay between the terminal's transmission of an EOM character and its transmission of the next line of data. See also *Prompt mode*.

vector

A straight line drawn between two points in terminal space.

Vector mode

One of the terminal's three *implicit command modes*. In Vector mode, the terminal interprets groups of incoming characters as xy-coordinates and draws vectors to the indicated locations.

volatile macro

A macro that is stored in volatile memory and is not retained when the terminal is powered off or reset. Compare to *nonvolatile macro*.

volatile memory

The part of the terminal's program memory that is erased each time the terminal is turned off. Compare to *nonvolatile memory*.

VT52 mode

One of the terminal's *host command modes*. VT52 mode allows the terminal to interpret VT52-style screen editing commands.

window

A user-defined rectangular area of terminal space. The terminal scales (and, if necessary, distorts) the image in the window to fill the terminal screen.

XY-coordinate parameter

A type of parameter used in TEK mode graphics commands issued from the host. The XY-coordinate parameter is a one-to-five-character string of ASCII characters that encodes the *terminal space coordinates* of a position in terminal space. See the beginning of Section 5 to see how to encode XY-coordinate parameters.

XY-coordinate report parameter

A type of *parameter* that the terminal uses when it reports locations in terminal space to the host. The XY-coordinate report is a one-to-five-character string of ASCII characters that the terminal uses to encode the *terminal space coordinates* of a position in terminal space. Compare to *4010 XY-report*.

Italicized words within a definition are terms defined elsewhere in this glossary.

INDEX

Keep these conventions in mind when using this index:

- Entries shown in all uppercase letters are command names.
- Where there is more than one page reference for an entry, the first reference is to the principal discussion of the topic.
- Italicized entries are Setup names and refer you only to the command description—for other references, see the index entry under the command's descriptive name.

- 4010 emulation, 4-6
4010 GIN, 4-31, 5-21, 5-85
4010 GIN Report, 5-21, 5-85
 decoding subroutine, 6-8
4010 HARDCOPY, 5-78, 2-18
4010 Status Report, 5-40, 5-86
4010 xy-report parameter, 5-80, 6-9
4014 line style, 5-77
4100-style commands, 4-2, 1-4, 1-6, 5-1—5-78
 entering, 5-8—5-10, 4-2
 terminating, 5-10
4100-style commands, syntax, 5-8—5-10
 command descriptions, 5-11—5-78
 encoded parameters for, 5-2—5-7
 error-codes for, B-2
 listed by function, on Section 5 *Commands* divider tab
 parameter types, 5-2—5-7
 table of, C-2
4104 Terminal, G-1
4644 Dot Matrix Printer, 2-14, 5-45
4691 Color Graphics Copier, 2-14, 5-45
4692 Color Graphics Copier, 2-14, 5-45
4695 Color Graphics Copier, 2-14, 5-45
- A**
Absolute Origin mode, 3-6, 3-34, 3-42, 5-50
ACURSOR, 5-45
ADE (ASCII decimal equivalent), 4-33, 5-2
alpha cursor, 3-3, 5-40, 5-45, 5-86
alpha cursor position
 4010 xy-coordinates, 2-12, 5-86
 current position, 3-3
 home position, 3-10, 3-26
 in 4010 Status Report, 2-12, 5-40, 5-86
 in Cursor Position Report, 2-12, 3-18
 row-and-column, 2-12, 3-18
Alpha mode, 4-3, 4-17
 entering, 5-23
 leaving, 5-14
Alphanumeric Controller, 1-5
alphanumeric cursor. *See* alpha cursor
alphanumeric memory, 1-4, 1-5
alphatext, in dialog area, 4-3, 4-6
 displaying, 5-20
 screen editing, 3-1—3-48
 underscoring, 3-8, 3-30
alphatext in dialog area, attributes, 3-30
 blinking, 3-8, 3-30
 character set, 3-7, 3-28, A-1
 color (index), 3-7, 5-53
 double height, 3-40
 double width, 3-40
 single width, 3-44
alphatext, in graphics area, 4-6
 and ENABLE DIALOG AREA command, 5-20
 and ENTER ALPHA MODE command, 5-23
 and SET GRAPHICS AREA WRITING MODE
 command, 5-60
 emulating 4010 terminal, 4-6, 5-20
alphatext in graphics area, attributes
 character color, 5-74
 font, 3-7, 3-28, 5-46, A-1
Alternate Keypad mode, 3-14, 3-41
ALU mode, 4-25, 5-13, 5-38, 5-41
ANSI commands, 3-12, 3-17—3-44
 canceling, 3-17, 3-37
 descriptions of, 3-17—3-44
 error codes for, B-8
 listed by function, Section 3 *ANSI/VT52 Commands* divider
 parameters, 3-15—3-16
 syntax conventions, 3-15—3-16
 table of, C-8
ANSI mode, 3-12, 1-6
 entering, 3-12, 3-29, 3-48, 5-43
ANSI X3.64 Standard, 3-2, 3-12, 1-6
ANSI, setting modes, 3-8, 3-27, 3-33—3-35
ANSI-TO-VT52 MODE (TEKANM), 3-38, 3-34
ANSI/VT52 Mode Interpreter, 1-5
ANSWERBACK, 5-46
answerback string, 2-10, 2-11, 3-22, 3-46, 5-46, 5-81
architecture, terminal, 1-4—1-7
array parameters
 integer, 5-6
 string, 5-6
array, runcode, 4-27, 5-41
ASCII/North American character set, A-2
aspect ratio, 4-10, 5-76
AUTOPRINT, 3-25, 2-18
AUTOREPEAT, 3-34, 3-38
AUTOWRAP, 3-34, 3-38

INDEX

B

B

- background color, dialog area, 3-7, 3-30, 5-53
 - character cell, 3-7, 3-30, 4-5, 5-53
 - dialog area, 3-7, 3-30, 4-5, 5-53
- background color, graphics area, 4-14
 - defining, 4-14, 5-75
 - reporting status, 5-84
- BACK SPACE (BS), 3-17
- BAUD RATE, 5-47, 2-2
- baud rate, 2-2
 - reporting, 5-83
 - setting, 2-2, 5-47
 - transmit rate limit, 2-2, 5-75
- BAUDRATE*, 5-47
- beam, electron, 4-7
- beam, pixel, 4-25, 5-68
- BEGIN PANEL BOUNDARY, 5-11, 4-23
- BEGIN PIXEL OPERATIONS, 5-13, 4-25
- BEGINPANEL*, 5-11
- BEL (BELL), 3-17
- bell, terminal, 3-17, 5-31
- bit packing scheme, 4-26
- bit planes, 4-9
- bits-per-pixel, 4-25, 5-13, 5-36, 5-41
- blanking. *See* erasing
- blinking
 - alpha cursor, 3-3, 5-45
 - alphatext, 3-8, 3-30
- bold display, 3-30
- boundary, panel, 4-23, 5-11
- Break key, 2-3, 5-47
- break signal, 2-3
 - duration, 5-47
- BREAKTIME*, 5-47, 2-3
- BS (BACKSPACE), 3-17
- buffer, dialog area, 3-2—3-6
 - erasing, 5-14
 - setting size, 3-2, 5-50
- buffering
 - input queue, 2-6, 5-70
 - output queue, 2-6
- buffering commands, 2-8
 - SET QUEUE SIZE, 5-70, 2-6
- bypass cancel character
 - and Bypass mode, 2-4, 5-23
 - specifying, 5-48
- Bypass mode, 2-4, 5-23
 - and bypass cancel character, 5-48
 - and reports, 5-81
 - entering, 5-23
 - exiting, 2-14, 5-14
- BYPASSCANCEL*, 5-48

C

- CAN (CANCEL), 3-17
- CANCEL*, 5-14
- CANCEL, 5-14
- Cancel key, 5-14
- Carriage Return
 - and CRLF command, 5-16
 - and ENABLE DIALOG AREA command, 5-20
 - and LFCR command, 5-30
 - and monochrome printers, 2-16, 5-64
 - and terminal display, 3-24, 3-34, 5-16, 5-30
 - See also* C_R
- CBT (CURSOR BACKWARD TAB), 3-17
- Centronics-style interface, 2-14
 - specifying, 5-45
- character
 - bypass cancel, 2-4, 5-23, 5-48
 - edit, 5-56
 - EOM (end-of-message), 2-8, 5-58
 - filler, 2-4, 5-28
 - key (in 4010 GIN Report), 5-85
 - key-execute, 4-34, 5-65
 - start, 2-7
 - stop, 2-7
 - character array. *See* string parameters
 - character background index, 5-53, 4-5
 - character cell, 3-30, 4-3, 5-53
 - character color, 3-30, 4-5, 5-53
 - character flagging, 2-7, 5-59
 - character height, graphtext, 4-21, 5-63
 - character input subroutine, 6-12
 - character output subroutine, 6-12
 - character parameters
 - host, 5-2
 - Setup, 5-6
 - character path, 4-21, 5-61
 - character report parameter, 5-79
 - character set (G0), 3-7, 3-28, 3-32, A-1
 - character set (G1), 3-7, 3-28, 3-36, A-1
 - character sets, default
 - alphatext, 3-28
 - graphtext, 4-21
 - character sets, for alphatext
 - designating G0 and G1, 3-7, 3-28, A-1
 - restoring, 3-10, 3-43
 - selecting, 3-7, 3-28, 3-32, 3-36, 5-46, A-1
 - character sets, for graphtext, 4-21
 - character sets, national, 3-7, 3-28
 - ASCII/North American, A-2
 - Danish/Norwegian, A-10
 - French, A-6
 - German, A-12
 - Swedish, A-8
 - United Kingdom, A-4
 - US (ASCII), A-2
 - character sets, special, 3-7, 3-28, A-1
 - Rulings, A-15
 - Supplementary, A-14
- CLEAR DIALOG SCROLL, 5-14

- CLEARDIALOG*, 5-14
- clipping
 - and SET WINDOW command, 5-76
 - outside of window, 4-10
- clusters, command
 - and HELP command, 5-28
 - and STATUS command, 5-78
- CMAP*, 5-72—5-73
- CODE*, 3-29
- code charts
 - ASCII/North American, A-2
 - Danish/Norwegian, A-10
 - French, A-6
 - German, A-12
 - Rulings, A-15
 - Supplementary, A-14
 - Swedish, A-8
 - United Kingdom, A-4
- color commands, 4-16
 - SELECT FILL PATTERN, 5-44
 - SET ALPHA CURSOR INDICES, 5-45
 - SET DIALOG AREA COLOR MAP, 5-51
 - SET DIALOG AREA INDEX, 5-53
 - SET LINE INDEX, 5-66
 - SET SURFACE COLOR MAP, 5-72—5-73, 5-84
 - SET TEXT INDEX, 5-74
- color cone, HLS, 4-12—4-13, E-1
- color coordinate system, HLS, 4-12—4-13, E-1
- color graphics copiers, 2-14—2-15, 2-17, 5-45
- color indices, in dialog area, 3-3, 3-7, 4-14
 - for alpha cursor, 3-3, 5-45
 - for alphanet, 3-7, 3-30, 5-53
 - for character background, 3-7, 3-30, 5-53
- color indices, in dialog area, default, 3-7, 5-51
- color indices, in graphics area, 4-14, 5-72—5-73, 5-75
 - for GIN cursor, 5-59
 - for lines, 5-66
 - for markers, 5-66, 5-19
 - for panel boundaries, 5-66
 - for pixels, 4-26, 4-27, 5-36—5-37, 5-38, 5-41
 - for text, 5-74
- color indices, in graphics area, default, 5-73
- color indices, in graphics area, reporting, 5-84
- color map
 - dialog area, 3-7, 5-51
 - graphics area, 4-14, 5-72—5-73
- color standard, Tektronix, 4-12—4-13, E-1
- colors
 - controlling from the keyboard, 1-2, 4-14
 - reversing in dialog area, 3-8, 3-30
 - See also* color indices
- colors. *See* color indices
- Column mode, 3-8, 3-27, 3-33, 3-34, 3-39
- COLUMNMODE*, 3-39
- command clusters
 - and HELP command, 5-28
 - and STATUS command, 5-78
- command conventions, 2-1, 3-1, 4-1, 5-1
 - ANSI mode, 3-15—3-16
 - TEK mode, 5-8—5-10
 - VT52 mode, 3-15—3-16
- command sets, 1-6, 1-7
 - ANSI, 3-12
 - TEK, 3-12
 - VT52, 3-14
- command syntax, selecting
 - from ANSI mode, 3-29, 3-38
 - from EDIT mode, 3-29, 3-12, 3-38, 4-2, 5-43
 - from VT52 mode, 3-47, 3-48
 - from TEK mode, 5-43
 - See also* syntax conventions
- commands
 - graphics, 4-1—4-31, 5-1—5-78
 - keyboard entry of, 2-9
 - screen editing, 3-1—3-48
- commands, ANSI mode
 - canceling, 3-17, 3-37
 - descriptions, 3-17—3-44
 - parameters, 3-15
 - syntax, 3-15—3-16
 - table of, C-8
- commands, TEK mode
 - descriptions, 5-11—5-78
 - omitting parameters, 5-10
 - parameters for, 5-1—5-7
 - syntax, 5-8—5-10
 - table of, C-2
 - terminating characters, 5-10
- commands, VT52 mode
 - descriptions, 3-45—3-48
 - syntax, 3-15—3-16
 - table of, C-14
- communications, 2-1—2-19
 - buffering, 2-6
 - copiers and printers, 2-14—2-18
 - handshaking, 2-7
 - host, 2-1—2-5
 - nonconventional, 2-9
 - reports, 2-11—2-13, 5-79—5-86
- communications, host
 - establishing, 2-1—2-5
 - interrupting, 2-3
 - pacing, 5-75, 2-2
 - Prompt mode, 2-8
 - security, 2-10
- communications, nonconventional, 2-19
- communications commands, buffering and handshaking
 - PROMPT MODE, 5-35, 2-8, 5-84
 - SET EOM CHARACTERS, 5-58, 2-8
 - SET FLAGGING MODE, 5-59, 2-7
 - SET PROMPT STRING, 5-70, 2-8
 - SET QUEUE SIZE, 5-70, 2-6
 - SET TRANSMIT RATE LIMIT, 5-75, 2-2

INDEX

C

communications commands, copier and printer, 2-14—2-17
 4010 HARDCOPY, 5-78, 2-18
 COPY, 5-15, 2-18
 HARDCOPY, 5-27, 2-15, 2-18
 MAP INDEX TO PRINT, 5-32, 2-16
 MC (MEDIA COPY), 3-25, 2-18
 SELECT COLOR HARDCOPY IMAGE DENSITY, 5-44, 2-15
 SELECT HARDCOPY INTERFACE, 5-45, 2-14
 SET COLOR COPIER REPAINT, 5-48, 2-15
 SET COPY SIZE, 5-49, 2-15
 SET DIALOG AREA HARDCOPY ATTRIBUTES, 5-52, 2-16
 SET IMAGE ORIENTATION, 5-65
 communications commands, host, 2-1—2-5
 ENTER BYPASS MODE, 5-23, 2-4
 IGNORE DELETES, 5-28, 2-2
 SET BAUD RATES, 5-47, 2-2
 SET BREAK TIME, 5-47, 2-3
 SET BYPASS CANCEL CHARACTER, 5-48, 2-4
 SET ECHO, 5-56, 2-3
 SET PARITY, 5-68, 2-2
 SET STOP BITS, 5-72, 2-2
 SET TRANSMIT RATE LIMIT, 5-75, 2-2
 communications commands, reports, 2-11—2-13
 REPORT 4010 STATUS, 5-40, 2-12, 5-86
 REPORT ERRORS, 5-38, 2-12, 5-82
 REPORT SYNTAX MODE, 3-26, 3-48, 5-39, 2-11
 REPORT TERMINAL SETTINGS, 5-39, 2-11, 5-82—5-85
 SET EOL STRING, 5-57, 2-8, 2-11, 5-81
 communications commands, security
 ENQUIRY, 3-22, 3-46, 5-22, 2-10, 2-11
 ENTER BYPASS MODE, 5-23, 2-4, 5-23
 SET ANSWERBACK STRING, 5-46, 2-10, 5-81
 SET ECHO, 5-56, 2-3
 communications modes
 Bypass mode, 2-4, 5-23, 5-48, 5-81
 Local mode, 2-9, 5-31
 Prompt mode, 2-8, 5-35
 communications settings
 baud rate, 2-2, 5-47
 data bits, 5-72, 2-2
 echo, 2-3, 3-34, 3-37, 5-56
 EOF string, 2-3, 5-57
 EOL string, 2-8, 2-11, 5-57
 flagging, 2-7, 5-59
 handshaking, 2-7
 parity, 2-2, 5-68
 start bit, 2-2, 5-72
 stop bit, 2-2, 5-72
 transmit delay, 2-8, 5-74
 transmit rate limit, 2-2, 5-75

communications settings, saving, 2-5
 compatibility with graphics programs, 1-3
 compatibility with other Tektronix terminals
 graphtext rotation, 5-62
 graphtext size, 5-63
 pixel dimensions, 4-25
 segment number, 5-71
 surface number, 5-72, 5-75
 compatibility with screen editing programs, 1-3
 control characters
 code chart, A-2, A-4, A-6, A-8, A-10, A-12
 displaying, 2-9, 5-71
 coordinates. *See* xy-coordinate parameters, xy-coordinate
 report parameters, 4010 xy-coordinate report parameters
 copier and printer commands, 2-14—2-17
 MAP INDEX TO PRINT, 5-32, 2-16
 SELECT COLOR HARDCOPY IMAGE DENSITY, 5-44, 2-15
 SELECT HARDCOPY INTERFACE, 5-45, 2-14
 SET COLOR COPIER REPAINT, 5-48, 2-15
 SET COPY SIZE, 5-49, 2-15
 SET DIALOG AREA HARDCOPY ATTRIBUTES, 5-52, 2-16
 SET HARDCOPY MONOCHROME ATTRIBUTES, 5-64, 2-16
 SET IMAGE ORIENTATION, 5-65, 2-15
 copier status, in 4010 Status Report, 2-12, 5-40, 5-86
 copiers and printers, 2-14—2-17, 5-45
 color copiers, 2-15, 2-17
 line endings for monochrome printers, 2-16, 5-64
 monochrome printers, 2-15, 2-17
 copiers and printers, specifying, 5-45
 COPY, 5-15
 COPY, 2-18, 5-15
 copy commands, 2-14—2-17
 4010 HARDCOPY, 5-78, 2-18
 COPY, 5-15, 2-18
 HARDCOPY, 5-27, 2-15, 2-18
 MC (MEDIA COPY), 3-25, 2-18
 copy. *See* hard copy
 CPR (CURSOR POSITION REPORT), 3-18, 2-12
^c_R (Carriage Return) character
 and CRLF command, 5-16
 and ENABLE DIALOG AREA command, 5-20
 and LFCR command, 5-30
 and monochrome printers, 2-16, 5-64
 and terminal display, 3-24, 3-34, 5-16, 5-30
 CR (CARRIAGE RETURN), 3-18
 CRLF, 5-16
 CRLF, 5-16
 crosshair cursor. *See* GIN cursor
 CTS (Clear to Send), 2-19, 2-7, 5-59
 CUB (CURSOR BACKWARD), 3-18
 CUD (CURSOR DOWN), 3-18
 CUF (CURSOR FORWARD), 3-19
 CUP (CURSOR POSITION), 3-19, 3-6

current line, 3-3
 current position, 3-3
 cursor, alpha, 3-3, 5-40, 5-85
 4010 xy-coordinates, 5-40, 5-86
 home position, 3-10, 3-26
 row and column address, 3-18, 3-20
 scrolling the dialog area, 3-4
 cursor, GIN, 4-31, 5-21, 5-59, 5-60
 cursor commands, ANSI
 BS (BACKSPACE), 3-17
 CBT (CURSOR BACKWARD TAB), 3-17
 CHT (CURSOR HORIZONTAL TAB), 3-17
 CR (CARRIAGE RETURN), 3-18
 CUB (CURSOR BACKWARD), 3-18
 CUD (CURSOR DOWN), 3-18
 CUF (CURSOR FORWARD), 3-19
 CUP (CURSOR POSITION), 3-19, 3-6
 CUU (CURSOR UP), 3-19
 HVP (HORIZONTAL AND VERTICAL POSITION), 3-23, 3-6
 IND (INDEX), 3-24
 LF (LINE FEED), 3-24
 NEL (NEXT LINE), 3-25
 RI (REVERSE INDEX), 3-26
 TEKRC (RESTORE CURSOR), 3-43
 TEKSC (SAVE CURSOR), 3-43
 cursor commands, VT52
 CURSOR DOWN, 3-45
 CURSOR LEFT, 3-45
 CURSOR RIGHT, 3-45
 CURSOR TO HOME, 3-45
 CURSOR UP, 3-45
 DIRECT CURSOR ADDRESS, 3-46
 REVERSE LINEFEED, 3-48
 CURSOR KEYS MODE (TEKCKM), 3-34, 3-39
 cursor position. *See* alpha cursor position, GIN cursor position
 CURSOR POSITION REPORT (CPR), 3-18, 2-12
 cursor speed, GIN, 4-31, 5-60
 CURSORKEYMODE, 3-34, 3-39
 curve, simulating, 4-18
 customizing the terminal, 4-2, 5-42
 CUU (CURSOR UP), 3-19

D

D Copy key, 3-25, 5-27
 D Eras key, 5-14
 DA (DEVICE ATTRIBUTES), 3-19
 DABUFFER, 5-50
 DACMAP, 5-51
 DAENABLE, 5-20
 DAINDEX, 5-53
 DALINES, 5-54
 DAMODE, 3-42, 5-55
 Danish/Norwegian character set, 3-28, A-10
 dashed line, 5-66, 5-77
 data bit, 5-72, 2-2
 data logging, 2-18, 3-25
 Data Terminal Ready (DTR), 2-7, 2-19, 5-59
 DAVISIBILITY, 5-54
 DC1/DC3 flagging, 2-7, 5-59
 DCH (DELETE CHARACTER), 3-20
 default color indices
 dialog area, 3-7, 5-51
 graphics area, 5-73, 5-75
 default copy size, 2-15, 5-49
 default parameter values
 for ANSI commands, 3-16
 for 4100-style commands, 5-8
 default parameter values, restoring, 5-25
 DEFINE, 5-16
 DEFINE MACRO, 5-16, 4-32—4-40
 DEFINE NONVOLATILE MACRO, 5-18, 4-32—4-40
 Delete (P_r) character, ignoring, 2-4, 5-28
 DELETE CHARACTER (DCH), 3-20
 DELETE LINE (DL), 3-20
 deletion commands, ANSI
 DCH (DELETE CHARACTER), 3-20
 DL (DELETE LINE), 3-20
 delimited string parameters, 5-7
 delimiter, in Setup
 ANSI commands, 3-15
 4100-style commands, 5-8
 delimiter, in host syntax, ANSI commands, 3-15
 DEVICE ATTRIBUTES (DA), 3-19
 DEVICE STATUS REPORT (DSR), 3-20, 2-12
 dialog area, 3-2—3-9, 4-3—4-6
 controlling, 3-3—3-9, 4-3—4-5
 copying, 2-18, 3-25, 5-27
 displaying messages, 4-4
 disabling, 5-20, 4-4, 4-6
 editing in Setup, 5-56
 enabling, 5-20, 4-4, 4-6
 printing, 2-18, 3-25, 5-27
 scrolling, 3-4
 setting the size, 3-2, 4-4, 5-54

INDEX

D

dialog area, display characteristics, 3-7—3-9
 background color, 3-7, 3-30, 4-5, 5-53
 character background color, 3-7, 3-30, 4-5, 5-53
 character blinking, 3-8, 3-30
 character color, 3-7, 3-30, 4-5, 5-53
 color map, 5-51
 number of lines, 3-2, 4-4, 5-54
 text underscoring, 3-8, 3-30
 transparency, 3-30
 visibility, 3-2, 4-4, 5-54
 writing mode, 5-55

dialog area commands, 3-9, 4-6
 CLEAR DIALOG SCROLL, 5-14
 ENABLE DIALOG AREA, 5-20, 4-4
 SET DIALOG AREA BUFFER SIZE, 5-50, 3-2, 4-4
 SET DIALOG AREA COLOR MAP, 5-51, 3-7
 SET DIALOG AREA HARDCOPY ATTRIBUTES, 5-52, 2-16
 SET DIALOG AREA INDEX, 5-53, 3-7, 4-5
 SET DIALOG AREA LINES, 5-54, 3-2, 4-4
 SET DIALOG AREA VISIBILITY, 5-54, 3-2, 4-4
 SET DIALOG AREA WRITING MODE, 5-55, 3-42

dialog area buffer. *See* dialog buffer

dialog buffer, 3-2, 4-4
 addressing the cursor, 3-6, 3-34, 3-42
 creating fixed regions, 3-6, 3-34, 3-44
 erasing, 3-26, 5-14
 scrolling, 3-4, 4-4

dialog buffer, attributes
 edit margins, 3-4—3-6, 3-34, 3-44
 fixed regions, 3-4—3-6, 3-34, 3-44, 5-50
 scrolling region, 3-4—3-6, 3-34, 3-44, 5-50
 size, 3-6, 4-4, 5-50
 width, 3-6, 3-34, 3-39

dialog buffer, memory used by, 4-42

Dialog key, 4-4, 5-54, 5-78

DIRECT CURSOR ADDRESS, 3-46

DISABLE MANUAL INPUT (DMI), 3-20

display, raster, 4-7

dither pattern, 5-44, F-1

DL (DELETE LINE), 3-20

DMI (DISABLE MANUAL INPUT), 3-20

dotted line, selecting, 5-66, 5-77

DOUBLE HEIGHT LINE (TEKDHL), 3-40

DOUBLE WIDTH LINE (TEKDWL), 3-40

DRAW MARKER, 5-19, 4-20

DRAW, 5-19, 4-18, 4-19

DRAW, 5-19

DSR (DEVICE STATUS REPORT), 3-20

[␣] character, 2-4, 5-28
 as filler character, 2-4, 5-28
 in encoded parameters, 5-28

DTR (Data Terminal Ready), 2-7, 2-19, 5-59

DTR/CTS flagging, 2-7, 5-59

E

ECH (ERASE CHARACTER), 3-21

ECHO, 5-56

echo, 2-3, 3-34, 3-37, 5-56

ED (ERASE IN DISPLAY), 3-21

edit characters, Setup
 in dialog area, 5-56
 in macros, 4-34

edit margins, 3-4—3-6, 3-34, 3-44

EDIT mode, 1-6, 3-12
 and TEK mode, 4-2, 5-43
 disabling key macros, 4-34
 entering, 3-12, 3-29, 3-48, 5-43

EDITCHARS, 5-56

editing, 3-2—3-14
 ANSI mode, 3-12, 1-6, 1-7
 EDIT mode, 3-12, 1-6, 1-7
 VT52 mode, 3-14, 1-6, 1-7

editing programs, 3-2, 1-3
 software compatibility, 1-3

EDITMARGINS, 3-44

EL (ERASE IN LINE), 3-21

electron beam, 4-9

EMI (ENABLE MANUAL INPUT), 3-22

ENABLE 4010 GIN, 5-21, 4-31

ENABLE DIALOG AREA, 5-20, 4-4

ENABLE KEY EXPANSION, 5-21, 3-12, 4-34

ENABLE MANUAL INPUT (EMI), 3-22

encoded integer parameters, 5-2—5-3
[␣] characters in, 5-28
 subroutine to encode, 6-4
 table of, D-1—D-6

encoded xy-coordinate parameters, 5-4—5-5
[␣] characters in, 5-28
 subroutine to encode, 6-6

END PANEL, 5-22, 4-23

end-of-file string (EOF), 2-3, 5-57
 in reports, 5-81

end-of-line string (EOL), 5-57, 2-8, 2-11, 5-81

end-of-message character (EOM), 2-8, 5-58

ENDPANEL, 5-22

ENQUIRY, 2-10, 2-11, 3-22, 3-46, 5-22

ENTER ALPHA MODE, 5-23, 4-3, 4-17

ENTER ALTERNATE KEYPAD MODE, 3-46, 3-14

ENTER ANSI MODE, 3-47

ENTER BYPASS MODE, 5-23, 2-4

ENTER GRAPHICS MODE, 3-47

ENTER MARKER MODE, 5-24, 4-17
 ENTER VECTOR MODE, 5-24, 4-17
 EOF (end-of-file) string, 2-3, 5-57
EOFSTRING, 5-57
 EOL (end-of-line) string, 2-8, 2-11, 5-57
 in reports, 5-81
EOLSTRING, 5-57
 EOM (end-of-message) character, 2-8, 5-58
EOMCHARS, 5-58
 ERASE CHARACTER (ECH), 3-21
 ERASE IN DISPLAY (ED), 3-21, 3-47
 ERASE IN LINE (EL), 3-21, 3-47
 erase index, 4-12
 ERASE TO END OF LINE, 3-47
 ERASE TO END OF SCREEN, 3-47
 erasure commands, ANSI
 ECH (ERASE CHARACTER), 3-21
 ED (ERASE IN DISPLAY), 3-21, 3-47
 EL (ERASE IN LINE), 3-21, 3-47
 erasure commands, VT52
 ERASE TO END OF LINE, 3-47
 ERASE TO END OF SCREEN, 3-47
 erasure commands, TEK
 CLEAR DIALOG SCROLL, 5-14
 PAGE, 5-34
 error codes, B-1—B-9, 2-12
 for 4100-style commands, B-2
 for ANSI commands, B-8
 severity levels, B-1
 Error Report, 5-82, 5-38
ERRORLEVEL, 5-58
 escape sequence, 4-2, 5-9
 even parity, 2-2, 5-68
 examples
 drawing lines in the graphics area, 6-13
 input/output, low-level, 6-12
 macros, using, 4-35—4-39
 parameters, encoding, 6-3—6-6
 pixel operations, using, 4-28—4-29
 prompts, user, how to build, 4-5
 RASTER WRITE command, encoding, 4-26
 reports, decoding, 6-7—6-11
 RUNLENGTH WRITE command, encoding, 4-27
 EXIT ALTERNATE KEYPAD MODE, 3-46
 EXIT GRAPHICS MODE, 3-47
EXPAND, 5-25
 EXPAND MACRO, 5-25, 5-21, 4-33

F
FACTORY, 5-25
 FACTORY, 3-11, 5-25
 factory defaults, 5-8
 restoring, 5-25
 fast copy, 2-15, 5-44
 FF (FORM FEED), 3-22
 interpretation, 2-16, 5-52
 fill pattern, 4-23, 5-44
 table of, F-1
 filler characters
 D_T , 2-4, 5-28
 $\text{F}_C?$, 2-4
FILLPATTERN, 5-44
 firmware version
 in Terminal Settings Report, 5-39, 5-82
 in STATUS, 5-78
 fixed regions, 3-4—3-6
 cursor-addressing, 3-34, 3-42
 edit margins, 3-44
FLAGGING, 5-59
 flagging, 2-7
 character, 5-59
 DTR/CTS, 5-59
 fonts, alphetext, 3-7, 3-28, 5-46, A-1
 See also character sets
 fonts, graphtext, 4-21
 FORM FEED (FF), 3-22
 interpretation, 2-16, 5-52
 French character set, 3-28, A-6
 function keys, 3-14
 Alternate Keypad mode, 3-46, 3-47
 Keypad Application mode, 3-41
 Keypad Numeric mode, 3-41
 TEKCKM (CURSOR KEYS MODE), 3-34, 3-39

G
 G Eras key, 5-34
 and ENABLE DIALOG AREA command, 5-20
 G0 character set, 3-7, 3-28, 3-32, A-1
 G1 character set, 3-7, 3-28, 3-36, A-1
GAMODE, 5-60
GCURSOR, 5-59
 German character set, 3-28, A-12
 GIN commands, 4-31
 ENABLE 4010 GIN, 5-21, 4-31
 SET GIN CURSOR COLOR, 5-59
 SET GIN CURSOR SPEED, 5-60, 4-31
 GIN cursor, attributes, 4-31
 color, 5-59
 speed, 5-60, 4-31
 GIN cursor position, 4-31
 moving, 5-71
 reporting, 5-40, 5-86
 updating, 5-34
 GIN reports, 4-31, 5-21, 5-85
 graphic rendition, 3-7, 3-30

INDEX

G

GRAPHIC TEXT, 5-26, 4-21
 graphics concepts, 4-1—4-42
 graphics area, 4-3
 copying, 2-18, 5-27
 controlling colors, 4-12—4-16
 displaying text in, 4-21—4-22
 erasing, 5-34
 using graphics primitives, 4-17—4-24
 using pixel operations, 4-25—4-30
 graphics area text commands, 4-22
 ENTER ALPHA MODE, 5-23
 GRAPHIC TEXT, 5-26
 SET ALPHATEXT FONT, 5-46
 SET GRAPHICS AREA WRITING MODE, 5-60
 SET GRAPHTEXT CHARACTER PATH, 5-61
 SET GRAPHTEXT ROTATION, 5-62
 SET GRAPHTEXT SIZE, 5-63
 SET TEXT INDEX, 5-74
 Graphics Controller firmware, 1-4, 1-5
 graphics input. *See* GIN
 graphics memory, 1-4, 1-5, 4-9
 coordinates, 4-9
 dimensions, 4-9
 off-screen, 4-9, 5-35, 5-69
 Graphics mode, VT52, 3-14
 entering, 3-47
 graphics position, 4-17
 home, 4-10
 moving, 5-33
 graphics primitives, 4-17
 alphatext, 4-22
 graphtext, 4-21
 markers, 4-20
 panels, 4-23—24
 vectors, 4-18
 graphics software compatibility, 1-3
 graphtext
 overwriting, 5-60
 replacing, 5-60
 writing, 5-26
 graphtext characters, attributes, 4-21
 character path, 5-61
 color, 5-74
 size, 5-63
 spacing, 5-63
 rotation, 5-62
 graphtext commands, 4-21
 GRAPHIC TEXT, 5-26
 SET GRAPHTEXT CHARACTER PATH, 5-61
 SET GRAPHTEXT ROTATION, 5-62
 SET GRAPHTEXT SIZE, 5-63
 SET TEXT INDEX, 5-74
 graphtext font, 4-21
 GSPEED, 5-60
 GTEXT, 5-26
 GTINDEX, 5-74
 GTPATH, 5-61
 GTROTATION, 5-62
 GTSIZE, 5-63

H

handshaking, 2-7
 flagging, 2-7, 5-59
 Prompt mode, 2-8, 5-35, 5-70
 handshaking commands, 2-8
 PROMPT MODE, 5-35, 2-8, 5-84
 SET EOM CHARACTERS, 5-58, 2-8
 SET FLAGGING MODE, 5-59, 2-7
 SET PROMPT STRING, 5-70, 2-8
 SET TRANSMIT RATE LIMIT, 5-75, 2-6
 HARDCOPY, 5-27, 2-15, 2-18
 hard copy, 4010 style, 2-18, 5-78
 hard copy, attributes
 image density, 2-15, 5-44
 image orientation, 2-15, 5-65
 line endings for monochrome printers, 2-16, 5-64
 monochrome print values, 2-16, 5-32
 position, 5-65
 repaint, 2-15, 5-48
 size, 2-15, 5-49
 hard copy, canceling, 5-14
 hard copy, creating, 2-15—2-18
 of dialog area, 2-18, 5-27
 of graphics area, 2-18, 5-27
 of host data, 2-18, 5-15
 of screen, 2-18, 5-27
 horizontal, 2-15, 5-65
 monochrome, 2-15—2-16, 5-32, 5-64
 vertical, 2-15, 5-65
 hard copy interface, 2-14, 5-45
 hardware flagging, 2-7, 5-59
 HCDAATTRIBUTES, 5-52
 HCDENSITY, 5-44
 HCINTERFACE, 5-45
 HCMAP, 5-32
 HCMONOCHROME, 5-64
 HCORIENT, 5-65
 HCREPAINT, 5-48
 HCSIZE, 5-49
 HELP, 5-28
 high parity, 5-68
 high-density copies, 5-44
 HLS color cone, 4-12, 4-13, E-1
 HLS color coordinate system, 4-12, 4-13, E-1
 home position
 dialog area, 3-10, 3-26, 5-20
 graphics, 4-10, 5-34
 HORIZONTAL AND VERTICAL POSITION (HVP), 3-6, 3-23
 horizontal scrolling, 3-4, 3-32, 3-36
 HORIZONTAL TAB (HT), 3-22
 HORIZONTAL TAB SET (HTS), 3-22
 host command modes, 1-6, 1-7, 3-12—3-14
 ANSI, 3-12
 EDIT, 3-12, 4-2, 5-43
 TEK, 3-12, 4-2
 VT52, 3-14

host command modes, selecting, 3-29, 3-48, 4-2, 5-43
host communications. *See* communications
host syntax
 ANSI mode commands, 3-15—3-16
 in macros, 4-35, 4-36
 TEK mode commands, 4-2, 5-8—5-10
 VT52 mode commands, 3-15—3-16
HT (HORIZONTAL TAB), 3-22
HTS (HORIZONTAL TAB SET), 3-22
hue (HLS), 4-12, 4-13, E-1
HVP (HORIZONTAL AND VERTICAL POSITION), 3-23, 3-6

I

ICH (INSERT CHARACTER), 3-23
IDENTIFY TERMINAL (TEKID), 3-40, 2-12
IDENTIFY, 3-48, 2-12
IGNORE DELETES, 2-4, 5-28
IGNOREDEL, 5-28
IL (INSERT LINE), 3-23
image density, 5-44
image orientation, 5-65
implicit command modes, 4-17—4-20
 Alpha mode, 4-3, 4-17, 5-22
 Marker mode, 4-17, 4-20, 5-24
 Vector mode, 4-17, 4-18—4-19, 5-24
implicit command modes, reporting status, 5-86
IND (INDEX), 3-24
Index 0, 4-12, 5-53
indices, color, in dialog area, 3-7
 alpha cursor, 3-3, 5-45
 alphatext, 3-7, 3-30
 background, 5-53
 blinking, 3-8, 3-30
 character, 3-7, 3-30
 character background, 3-7, 3-30, 5-53
 color map, 3-7, 3-51
 default, 5-51
indices, color, in graphics area, 4-12—4-16
 background, 5-75
 color map, 5-72—5-73
 default, 5-73
 GIN cursor, 5-59
 lines, 5-66, 5-19
 markers, 5-66, 5-19
 panel boundaries, 5-66
 pixels, 4-26, 4-27, 5-36—5-37, 5-38, 5-41
 reporting, 5-84—5-85
 setting, 4-14, 5-72—5-73, 5-75
 text, 5-74
initialization subroutine, 6-2
input queue, 2-6
 flushing, 5-14
 selecting size, 5-70
input queue, memory used by, 4-42
input/output subroutines, low-level, 6-12
inquiry codes, terminal settings, 2-11, 5-39, 5-82
INSERT CHARACTER (ICH), 3-23
INSERT LINE (IL), 3-23
INSERT/REPLACE MODE (IRM), 3-24, 3-34

insertion commands, ANSI
 ICH (INSERT CHARACTER), 3-23
 IL (INSERT LINE), 3-23
INSERTREPLACE, 3-24, 3-34
integer array parameters
 encoding subroutine, 6-4
 host, 5-6
 Setup, 5-7
integer array report parameters, 5-81
integer parameters, host, ANSI mode, 3-15
integer parameters, host, TEK mode
 \uparrow characters in, 5-28
 encoded, table of, D-1—D-6
 encoding, 5-2—5-3
 encoding subroutine, 6-4
 host, 5-2—5-3
integer parameters, Setup
 ANSI mode, 3-15
 TEK mode, 5-6
integer report parameters, 5-79
 decoding subroutine, 6-11
Interactive Color Interface, 4-14, 5-51, 5-73
international character sets, 3-7, 3-28, A-1
 ASCII/North American, A-3
 Danish, A-11
 French, A-7
 German, A-13
 Swedish, A-9
 United Kingdom, A-5
IRM (INSERT/REPLACE MODE), 3-24, 3-34
ISO 6429 Standard, 3-2, 1-4

J

Joydisk
 as GIN device, 4-31
 scrolling the dialog area, 3-4

K

KAM (KEYBOARD ACTION MODE), 3-34
key combinations, 4-33
 ASCII/North American keyboard, A-3
 Danish/Norwegian keyboard, A-11
 French keyboard, A-7
 German keyboard, A-13
 Swedish keyboard, A-9
 United Kingdom keyboard, A-5
key-execute character, 4-34
 in examples, 4-35, 4-37, 4-38
 in macros expanded from host, 4-34
 specifying, 5-65
key macro, 4-33—4-34, 4-39, 5-17
 See also macros
key specifier parameter, 5-7, 5-17
KEYBOARD ACTION MODE (KAM), 3-34
keyboards, 3-7, 3-28, A-1
 controlling characteristics, 3-7—3-9
 disabling, 3-20, 3-34, 5-31
 enabling, 3-22, 3-34, 5-14, 5-31
 locking and unlocking, 3-20, 3-22, 3-34, 5-31
 selecting character set, 3-7, 3-28, 4-21, 5-46, A-1

INDEX

K

keyboards, illustrated
 ASCII/North American, A-3
 Danish, A-11
 French, A-7
 German, A-13
 Swedish, A-9
 United Kingdom, A-5
KEYEXCHAR, 5-65
KEYEXPAND, 5-21
KEYPAD APPLICATION MODE (TEKKPAM), 3-41, 3-14
keypad modes, ANSI, 3-14
 application, 3-41
 numeric, 3-41
keypad modes, VT52, 3-14
 alternate, 3-46, 3-47
 programming codes, 3-46
KEYPADMODE, 3-41, 3-46, 3-47
keys
 defining from the keyboard, 4-33, 5-29, 5-30
 See also key combinations
keys, terminal
 Break key, 2-3, 5-31, 5-47
 Cancel key, 5-14, 5-31
 D Copy key, 2-18, 3-25, 5-27
 D Eras key, 5-14
 Dialog key, 4-4, 5-54, 5-78
 G Eras key, 5-20, 5-34
 S Copy key, 2-18, 5-27, 5-78
 S Eras key, 5-20
keyword parameter, Setup
 4100-style, 5-7
 ANSI mode, 3-15

L

LEARN, 5-29
LEARN, 5-29, 4-32
LEARN NONVOLATILE, 5-30, 4-32, 4-36—4-39
LF (Line Feed) character
 and monochrome printers, 2-16, 5-64
 and terminal display, 3-24, 3-34, 5-16, 5-30
LF (LINE FEED), 3-24, 3-34
LFCR, 5-30
LFCR, 5-30, 3-24, 3-34
lightness (HLS), 4-12
limit, transmit rate, 2-2, 5-75
line, current, 3-3
LINE FEED (LF), 3-24, 3-34
Line Feed (LF) character
 and monochrome printers, 2-16, 5-64
 and terminal display, 3-24, 3-34, 5-16, 5-30
LINEFEED/NEWLINE MODE (LNM), 3-24, 3-34
line of data
 in output queue, 2-8
 in Prompt mode, 2-8
 in reports, 5-81
line style, 4-18, 5-66, 5-77
LINEINDEX, 5-66
lines, 4-18—4-19
 drawing, 4-18—4-19, 5-19
 example, drawing, 6-13

lines, attributes
 color index, 4-18, 5-66
 line style, 4-18, 5-66, 5-77
LINESTYLE, 5-66
literal characters, 4-34, 5-16, 5-56
LNM (LINEFEED/NEWLINE MODE), 3-24, 3-34
LOCAL, 5-31
LOCAL, 5-31, 2-9
local echo, 2-3, 3-34, 3-37, 5-56
Local mode, 2-9, 5-31
LOCK KEYBOARD, 5-31
low-density copies, 5-44

M

macro charts, A-3—A-13
macro destination, changing, 4-34, 5-65
MACRO STATUS, 5-32
macros, 4-32—4-40
 and key-execute character, 4-34, 5-65
 defining from the host, 4-33, 5-16—5-18
 defining from the keyboard, 4-33, 5-29—5-30
 deleting, 4-40, 5-16—5-18
 displaying definitions, 5-32
 examples, 4-35—4-39
 executing from the host, 4-33, 5-25
 executing from the keyboard, 4-33, 5-25
 memory used by, 4-32, 4-42
 nesting, 5-25
 programming keys, 4-33—4-34, 4-39, 5-16—5-18, 5-29, 5-30
 storing in memory, 4-32, 5-42
 switching destination, 4-34, 5-65
 using locally, 4-34, 5-65
macros, key, 4-33—4-34, 4-39, 5-17, 5-29, 5-30
MACROSTATUS, 5-32
manuals, list of related, 1-1
MAP INDEX TO PRINT, 5-32, 2-16
margins, edit, 3-4—3-6, 3-34, 3-44, 5-50
MARKER, 5-19
marker commands, 4-20
 DRAW MARKER, 5-19
 ENTER MARKER MODE, 5-24
 SET MARKER TYPE, 5-67
Marker mode, 4-17, 4-20
 and panels, 5-11
 entering, 5-24
 exiting, 4-17, 5-14
markers, 4-20
 drawing, 5-19
 selecting, 5-67
 selecting color, 5-66
markers
 and SET GRAPHICS AREA WRITING MODE command, 5-60
 in panels, 5-11
 predefined, 4-20, 5-67
MARKERTYPE, 5-67
MEDIA COPY (MC), 2-18, 3-25
memory, available, reporting, 2-11, 4-42, 5-39, 5-82, 5-83
memory, out-of-memory error, 4-42, 5-29

memory, types, 4-41
 graphics, 4-9
 nonvolatile, 4-41, 5-42
 off-screen, 4-9, 5-35, 5-69
 program, 4-41
 volatile, 4-41

memory, use of by terminal features, 4-42

modes, ANSI, dialog display and keyboard, 3-7—3-9
 resetting, 3-27
 setting, 3-33—3-35

modes, communication
 Bypass, 2-4, 5-23, 5-48, 5-81
 Local, 2-9, 5-31
 Prompt, 2-8, 5-35, 5-58, 5-70

modes, host command, 1-4, 1-5
 ANSI, 3-12
 EDIT, 3-12, 4-2, 5-43
 TEK, 3-12, 4-2
 VT52, 3-14, 3-15—3-16

modes, host command
 reporting, 2-11, 3-26, 3-48
 selecting, 3-12, 3-29, 3-48, 5-43

modes, implicit command, 4-17
 Alpha, 4-3, 5-23
 Marker, 4-20, 5-24
 Vector, 4-18—4-19, 5-24

monochrome printer commands
 MAP INDEX TO PRINT, 5-32, 2-16
 SET HARDCOPY MONOCHROME ATTRIBUTES, 5-64, 2-16

monochrome printers, 2-15, 5-32, 5-64
 line endings for, 2-16, 5-64

monochrome printers, selecting, 5-45

MOVE, 5-33
 MOVE, 5-33, 4-18, 4-19

N

NEL (NEXT LINE), 3-25

no parity, 5-68

nonvolatile command settings, saving, 5-42
 ANSI command settings, 3-15
 answerback string, 2-10
 copier and printer command settings, 2-14
 4100-style command settings, 4-2, 5-10

nonvolatile macros, 4-32, 4-36—4-39, 5-17, 5-18
 creating, 5-18, 5-30, 4-32
 deleting, 5-18, 5-30, 4-32
 saving, 5-42, 4-32

nonvolatile memory, 4-41, 5-42
 deleting macros in, 4-42, 5-18, 5-30
 lifespan of, 5-42
 saving macros in, 5-18, 4-32, 5-42

North American character set, 3-28, A-2

Norwegian character set, 3-28, A-10

number of pages, hard copy, 5-52

numeric keypad, 3-14, 3-41
 programming codes, 3-41

Numeric Keypad mode, 3-14, 3-41

NVDEFINE, 5-18
NVLEARN, 5-30
NVSAVE, 5-42

O

odd parity, 5-68

off-screen memory, 4-9, 5-35, 5-69

on-screen memory, 4-9

opcode, 4-2, 5-8

ORIGINMODE, 3-34, 3-42

Origin mode, 3-6, 3-34, 3-42, 5-50
 restoring, 3-10, 3-43
 setting, 3-42

output queue, 2-6
 controlling with Prompt mode, 2-8
 flushing, 5-14

OVERSTRIKE/REPLACE MODE (TEKORM), 3-34, 3-42

P

PAGE, 5-34
 and ENABLE DIALOG AREA command, 5-20

page feed. *See* Form Feed

panel commands, 4-24
 BEGIN PANEL BOUNDARY, 5-11
 END PANEL, 5-22
 SELECT FILL PATTERN, 5-44

panels, 4-17, 4-23—4-24
 beginning, 4-23, 5-11
 ending, 4-23, 5-22
 filling, 4-23, 5-44

panels, attributes, 4-23—4-24
 and Marker mode, 4-23, 5-11
 and markers, 5-11
 boundary, 4-24, 5-11
 fill patterns (color), 5-44
 multiple boundaries, 4-24

panels, memory used by, 4-42

parameter encoding
 integer, 5-2—5-3
 integer array, 5-6
 xy-coordinate, 5-4—5-5

parameter-encoding subroutines, 6-3—6-6
 integer, 6-4
 integer array, 6-4
 string, 6-5
 xy-coordinate, 6-6

parameter types
 host, 5-1—5-6
 report, 5-79—5-81
 Setup, 5-6—5-7

parameter values
 displaying current status, 5-78
 saving current settings, 2-10, 2-14, 3-15, 4-2, 5-10, 5-42

parameters, host, ANSI mode, 3-15
 prefixed, 3-27, 3-31, 3-33

parameters, host, TEK mode
 character, 5-2
 integer, 5-2—5-3
 integer array, 5-6
 string, 5-6
 xy-coordinate, 5-4—5-5

INDEX

P

- parameters, report
 - 4010 xy-report parameter, 5-80, 6-9
 - character, 5-79
 - integer, 5-79, 6-11
 - integer array, 5-81
 - string, 5-81, 6-10
 - xy-coordinate, 5-80
 - parameters, Setup, 4100-style
 - character, 5-6
 - delimited string, 5-7
 - integer, 5-6
 - integer array, 5-7
 - key specifier, 5-7
 - keyword, 5-7
 - small integer, 5-6
 - string, 5-7
 - xy-coordinate, 5-6
 - parameters, Setup, ANSI, 3-15
 - keyword, 3-15
 - integer, 3-15
 - prefixed, 3-27, 3-31, 3-33
 - parameters, Setup, omitting, 5-10
 - PARITY*, 5-68
 - parity, 2-2, 5-68
 - path, graphtext character, 4-21, 5-61
 - peripheral devices. *See* copiers and printers
 - pixel beam position, 4-25, 5-68
 - pixel commands, 4-30
 - BEGIN PIXEL OPERATIONS, 5-13, 4-25
 - PIXEL COPY, 5-34—5-35, 4-27
 - RASTER WRITE, 5-36—5-37, 4-26
 - RECTANGLE FILL, 5-38, 4-27
 - RUNLENGTH WRITE, 5-41, 4-27
 - SET PIXEL BEAM POSITION, 5-68, 4-25
 - SET PIXEL VIEWPORT, 5-69, 4-25
 - PIXEL COPY, 5-34—5-35, 4-27
 - pixel viewport, 4-25, 5-69
 - pixels, 4-7, 4-25—4-30
 - copying, 5-34—5-35, 4-27
 - creating mirror images, 5-35
 - erasing, 4-25, 5-13, 5-35
 - example, 4-28—4-29
 - using the ` character, 5-36
 - writing, 4-26—4-27, 5-35, 5-36—5-37, 5-38, 5-41
 - polygon. *See* panel
 - power-up condition
 - and implicit command modes, 5-23
 - resetting, 5-40
 - program memory, 4-41
 - reporting, 5-83
 - programming examples
 - command encoding, 6-3
 - macros, 4-35—4-39
 - parameters, encoding, 6-3—6-6
 - pixel operations, 4-28—4-29
 - prompts, user, how to build, 4-5
 - RASTER WRITE command, encoding, 4-26
 - reports, decoding, 6-7—6-11
 - RUNLENGTH WRITE command, encoding, 4-27
 - sample program, 6-13
 - Prompt mode, 2-8, 5-35, 5-70
 - entering, 5-35
 - exiting, 5-35, 5-14
 - reporting, 5-84
 - transmit delay, 5-74
 - PROMPTMODE*, 5-35
 - prompt string, 2-8, 5-70
 - PROMPTSTRING*, 5-70
 - prompts, how to build, 4-5
 - PXBEGIN*, 5-11
 - PXCOPY*, 5-34
 - PXPOSITION*, 5-68
 - PXRASTERWRITE*, 5-36
 - PXRECTANGLE*, 5-38
 - PXRUNLENGTHWRITE*, 5-41
 - PXVIEWPORT*, 5-69
- ### Q
- quadruples, 4-14, 5-51, 5-72
 - queue, input, 2-6
 - flushing, 5-14
 - setting size, 5-70
 - queue, output, 2-6, 2-8
 - QUEUESIZE*, 5-70
- ### R
- RAM. *See* memory
 - raster display, 4-7—4-8
 - RASTER WRITE, 5-36—5-37, 4-26
 - example, 4-26
 - RECTANGLE FILL, 5-38, 4-27
 - reduced copy image, 5-49
 - Relative Origin mode, 3-6, 3-34, 3-42, 5-50
 - remote echo, 2-3, 3-34, 3-37, 5-56
 - repainting copy image, 2-15, 5-48
 - report commands, 4100-style, 2-11—2-13
 - ENABLE 4010 GIN, 5-21, 4-31
 - ENQUIRY, 5-22, 2-10, 2-11
 - REPORT ERRORS, 5-38, 2-12, 5-82
 - REPORT SYNTAX MODE, 2-11, 3-26, 3-48, 5-39
 - REPORT TERMINAL SETTINGS, 5-39, 2-11, 5-82—5-85
 - REPORT 4010 STATUS, 5-40, 2-12, 5-86
 - SET EOL STRING, 5-57, 2-8, 2-11, 5-81
 - report commands, ANSI, 2-11, 2-12
 - CPR (CURSOR POSITION REPORT), 3-18, 2-12, 3-20
 - DA (DEVICE ATTRIBUTES), 3-19, 2-12
 - DSR (DEVICE STATUS REPORT), 3-20, 2-12
 - ENQUIRY, 3-22, 2-10, 2-11
 - REPORT SYNTAX MODE, 3-26, 2-11
 - TEKID (IDENTIFY TERMINAL), 3-40, 2-12
 - report commands, VT52
 - ENQUIRY, 3-46, 2-10, 2-11
 - IDENTIFY, 3-48, 2-12
 - REPORT SYNTAX MODE, 3-38, 2-11
 - report decoding subroutine, 4010 GIN Report, 6-8
 - report input subroutine, 6-7

report parameter decoding subroutines

- 4010 xy-report parameter, 6-9
- integer report parameter, 6-11
- string report parameter, 6-10

report parameters, 5-79—5-81, 2-11

- 4010 xy report, 5-80, 6-9
- character, 5-79
- integer, 5-79, 6-11
- integer array, 5-81
- string, 5-81, 6-10
- xy-coordinate, 5-80

report syntax

- ANSI mode reports, 3-16
- TEK mode reports, 5-79—5-86

REPORT ERRORS, 5-38, 2-12, 5-82

REPORT SYNTAX MODE, 3-26, 3-48, 5-39, 2-11

REPORT TERMINAL SETTINGS, 5-39, 2-11, 5-82—5-85

REPORT 4010 STATUS, 5-40, 2-12, 5-86

reports, ANSI mode, 2-11, 2-12

- cursor position, 2-12, 3-18, 3-20
- device attributes, 2-12, 3-19
- device status, 2-11, 3-18, 3-20
- screen editing characteristics, 2-12, 3-19
- Syntax mode, 2-11, 3-26
- terminal type, 2-12, 3-19, 3-40

reports, requesting, 2-11—2-13

reports, TEK mode, 2-11—2-12, 5-81—5-86

- 4010 GIN report, 4-31, 5-21, 5-85
- 4010 Status, 5-86, 5-40
- Cursor Position, 5-86, 2-12, 5-40
- Error, 5-81, 2-12, 5-38
- Syntax Mode, 2-11, 5-39
- Terminal Settings, 2-11, 5-82—5-85, 5-39

reports, VT52 mode

- Syntax mode, 2-11, 3-48
- terminal type, 2-12, 3-48

Request to Send (RTS), 2-19

RESET, 5-40

RESET, 5-40

RESET MODE (RM), 3-27, 3-8, 3-33, 3-34—3-35

- parameter values, 3-35

RESET TO INITIAL STATE (RIS), 3-26, 3-10

RESTORE CURSOR (TEKRC), 3-43, 3-10

REVERSE INDEX (RI), 3-26

REVERSE LINEFEED, 3-48

reverse video display, 3-8, 3-30

RI (REVERSE INDEX), 3-26

RIS (RESET TO INITIAL STATE), 3-26, 3-10

RM (RESET MODE), 3-27, 3-8, 3-33, 3-34—3-35

- parameter values, 3-35

rotation angle, graphtext, 4-21, 5-62

RS-232 communications. *See* communications

RS-232 port, nonconventional uses, 2-19

Rulings character set, 3-28, A-15

runcode, 4-27, 5-41

RUNLENGTH WRITE, 5-41, 4-27

- example, 4-27

S

S Copy key, 2-18, 5-27, 5-78

S Eras key, and ENABLE DIALOG AREA command, 5-20

sample program, 6-13

saturation (HLS), 4-12—4-13, E-1

SAVE CURSOR (TEKSC), 3-43, 3-10

SAVE NONVOLATILE PARAMETERS, 5-42

- and 4100-style command settings, 4-2, 5-10
- and ANSI-style command settings, 3-15
- and answerback string, 2-10
- and copier and printer command settings, 2-14

screen display areas, 4-2

screen editing, 3-1—3-14

- ANSI mode, 3-12, 1-6
- EDIT mode, 3-12, 1-6
- VT52 mode, 3-12, 1-6

SCREEN MODE (TEKSCNM), 3-34, 3-43

SCREENMODE, 3-34, 3-43

scrolling, 3-4—3-6

- and cursor, 3-4
- and Joydisk, 3-4
- horizontal, 3-32, 3-36
- vertical scrolling, 3-29, 3-37

scrolling commands, ANSI, 3-4

- SD (SCROLL DOWN), 3-29
- SL (SCROLL LEFT), 3-32
- SR (SCROLL RIGHT), 3-36
- SU (SCROLL UP), 3-37

scrolling region, 3-4—3-6

- cursor-addressing, 3-34, 3-42
- edit margins, 3-4—3-6, 3-44

SCS (SELECT CHARACTER SET), 3-28, 3-7, 4-21

SD (SCROLL DOWN), 3-29, 3-4

Security, 2-10

- answerback string, 3-22, 3-46, 5-22, 5-46, 5-81
- Bypass mode, 2-4, 5-23, 5-48, 5-81
- ENQUIRY command, 3-22, 3-46, 5-22
- SET ANSWERBACK STRING command, 5-46

security commands, 2-10

- ENQUIRY, 2-10, 2-11, 3-22, 3-46, 5-22
- ENTER BYPASS MODE, 5-23, 2-4, 2-10
- SET ANSWERBACK STRING, 5-46, 2-10, 5-81
- SET ECHO, 5-56, 2-3, 2-10

segment position, 5-71

SELECT CHARACTER SET (SCS), 3-28, 3-7, A-1

SELECT CODE, 3-29, 3-48, 5-43, 3-12

SELECT COLOR HARDCOPY IMAGE DENSITY, 5-44, 2-15

SELECT FILL PATTERN, 5-44, 4-23

SELECT GRAPHIC RENDITION (SGR), 3-30, 3-7—3-8

SELECT HARDCOPY INTERFACE, 5-45, 2-14

SELECTCHARSET, 3-28

SEND/RECEIVE MODE (SRM), 3-37, 3-34

SET ALPHA CURSOR INDICES, 5-45, 3-7

SET ALPHATEXT FONT, 5-46, 4-21

SET ANSWERBACK STRING, 5-46, 2-10, 3-22, 3-46, 5-22, 5-81

SET BAUD RATES, 5-47, 2-2, 5-83

SET BREAK TIME, 5-47, 2-3

SET BYPASS CANCEL CHARACTER, 5-48, 2-4, 5-23

SET COLOR COPIER REPAINT, 5-48, 2-15

Set Color function. *See* Interactive Color Interface

SET COPY SIZE, 5-49, 2-15

INDEX

S

- SET DIALOG AREA BUFFER SIZE, 5-50, 3-2, 4-4
- SET DIALOG AREA COLOR MAP, 5-51, 3-7
- SET DIALOG AREA HARDCOPY ATTRIBUTES, 5-52, 2-16
- SET DIALOG AREA INDEX, 5-53, 3-7, 4-4
- SET DIALOG AREA LINES, 5-54, 3-2, 4-4
- SET DIALOG AREA VISIBILITY, 5-54, 3-2, 4-4
- SET DIALOG AREA WRITING MODE, 5-55, 3-42
- SET ECHO, 5-56, 2-3
- SET EDIT CHARACTERS, 5-56, 4-34
- SET EOF STRING, 5-57, 2-3
- SET EOL STRING, 5-57, 2-8, 2-11, 5-81
- SET EOM CHARACTERS, 5-58, 2-8
- SET ERROR THRESHOLD, 5-58, B-1
- SET FLAGGING MODE, 5-59, 2-7
- SET GIN CURSOR COLOR, 5-59
- SET GIN CURSOR SPEED, 5-60, 4-31
- SET GRAPHICS AREA WRITING MODE, 5-60, 4-22
- SET GRAPHTEXT CHARACTER PATH, 5-61, 4-21
- SET GRAPHTEXT ROTATION, 5-62, 4-21
- SET GRAPHTEXT SIZE, 5-63, 4-21
- SET HARDCOPY MONOCHROME ATTRIBUTES, 5-64, 2-16
- SET IMAGE ORIENTATION, 5-65, 2-15
- SET KEY EXECUTE CHARACTER, 5-65, 4-34
- SET LINE INDEX, 5-66, 4-18
- SET LINE STYLE, 5-66, 4-18
- SET MARKER TYPE, 5-67, 4-20
- SET MODE (SM), 3-33—3-35, 3-8, 3-27
 - parameter values, 3-35
- SET PARITY, 5-68, 2-2
- SET PIXEL BEAM POSITION, 5-68, 4-25
- SET PIXEL VIEWPORT, 5-69, 4-25
- SET PROMPT STRING, 5-70, 2-8
- SET QUEUE SIZE, 5-70, 2-6, 4-42
- SET SEGMENT POSITION, 5-71
- SET SNOOPY MODE, 5-71
- SET STOP BITS, 5-72, 2-2
- SET SURFACE COLOR MAP, 5-72—5-73, 4-14, 5-84
- SET TAB STOPS, 5-73
- SET TEXT INDEX, 5-74, 4-22
- SET TRANSMIT DELAY, 5-74, 2-8
- SET TRANSMIT RATE LIMIT, 5-75, 2-2
- SET VIEW ATTRIBUTES, 5-75, 4-12, 4-22
- SET WINDOW, 5-76, 4-10
- SET 4014 LINE STYLE, 5-77
- Setup syntax
 - 4100-style, 4-2, 5-8—5-10
 - ANSI, 3-15—3-16
 - edit characters for dialog area, 5-56
- SGPOSITION*, 5-71
- SGR (SELECT GRAPHIC RENDITION), 3-30, 3-7—3-8
- SHIFT IN (SI), 3-32, 3-7, 3-28, A-1
- SHIFT OUT (SO), 3-36, 3-7, 3-28, A-1
- SINGLE WIDTH LINE (TEKSWL), 3-44
- SL (SCROLL LEFT), 3-4, 3-32
- SM (SET MODE), 3-33—3-35, 3-8, 3-27
 - parameter values, 3-35
- small integer parameter, Setup, 5-6
- SNOOPY*, 5-71
- Snoopy mode, 2-9, 5-71
 - exiting, 5-71, 5-14
- SO (SHIFT OUT), 3-36, 3-7, 3-28, A-1
 - software compatibility, 1-3
 - software flagging, 2-7
- Space character, 3-34, 3-42, 3-42
- special inquiry codes, 2-11, 5-39, 5-82
- SR (SCROLL RIGHT), 3-36, 3-4
- SRM (SEND/RECEIVE MODE), 3-34, 3-37
- standards
 - ANSI X3.64, 3-2, 1-3, 1-4
 - ISO 6429, 3-2, 1-4
- start bits, 2-2, 5-72
- start character, 2-7
- starting page, for hard copies, 2-16, 5-52
- STATUS*, 5-78
- status, reporting, 2-11, 2-12, 5-78
 - 4010 terminal, 2-12, 5-40, 5-86
 - command clusters, 2-11, 5-78
 - copier, 2-12, 5-40, 5-86
 - cursor position, 2-12, 3-18, 5-40, 5-86
 - firmware version, 2-11, 5-39, 5-78
 - implicit command mode, 2-11, 5-86
 - memory, 2-11, 5-39, 5-78
 - parameter values, 2-11, 5-78, 5-82—5-85
 - Prompt mode, 5-84
 - single command, 2-11, 5-78
 - terminal model, 2-11, 5-39, 5-78
- stop bits, 2-2, 5-72
- STOPBITS*, 5-72
- string parameters, host, 5-6
 - count, 5-6
 - encoding subroutine, 6-5
- string parameters, Setup, 5-7
 - delimited, 5-7
- string report parameters
 - count, 5-81
 - decoding subroutine, 6-10
 - empty string, 5-81
- SU (SCROLL UP), 3-37, 3-4
- SUB (SUBSTITUTE), 3-37
- SUBSTITUTE (SUB), 3-37
- submodes, VT52, 3-14
 - Alternate Keypad, 3-46, 3-47
 - Graphics, 3-47
- subroutine examples
 - command parameter encoding, 6-3—6-7
 - initialization, 6-2
 - low-level input/output support, 6-12
 - terminal report decoding, 6-7—6-11
- Supplementary character set, 3-28, A-14
- surface, setting color for, 5-75
- surface color map, 5-72—5-73
 - reporting, 5-84
- Swedish character set, 3-28, A-8
- syntax conventions
 - ANSI mode, 3-15—3-16
 - TEK mode, 5-8—5-10
 - VT52 mode, 3-15—16
- syntax mode. *See* host command mode

T

- TAB CLEAR (TBC), 3-37
- tab stops, 3-22, 3-37, 5-73
- tabbing commands, ANSI
 - CBT (CURSOR BACKWARD TAB), 3-17
 - CHT (CURSOR HORIZONTAL TAB), 3-17
 - HT (HORIZONTAL TAB), 3-22
 - HTS (HORIZONTAL TAB SET), 3-22
 - TBC (TAB CLEAR), 3-37
 - VT (VERTICAL TAB), 3-44
- tabbing command, 4100-Style, SET TAB STOPS, 5-73
- TABS, 5-73
- TBC (TAB CLEAR), 3-37
- TEK mode, 4-2, 3-12
 - and terminal architecture, 1-4, 1-6
 - command descriptions, 5-11—5-78
 - and EDIT mode, 5-43, 3-12, 4-2
 - syntax for, 5-8—5-10
- TEK mode commands
 - and terminal architecture, 1-4, 1-6
 - and graphics commands, 4-2
 - encoded parameters for, 5-1—5-7
 - entering, 5-8—5-10, 3-12, 4-2
 - listed by function, Section 5 *Commands* divider
 - syntax conventions, 5-8—5-10
 - table of, C-2
 - terminating character, 5-10
 - using, 5-8—5-10
- TEK Mode Interpreter, 1-4, 1-5
- TEKANM (ANSI-TO-VT52 MODE), 3-34, 3-38
- TEKARM (AUTOREPEAT MODE), 3-34, 3-38
- TEKAWM (AUTOWRAP MODE), 3-34, 3-38
- TEKCKM (CURSOR KEYS MODE), 3-34, 3-39
- TEKCOLM (COLUMN MODE), 3-34, 3-39
- TEKDHL (DOUBLE HEIGHT LINE), 3-40
- TEKDWL (DOUBLE WIDTH LINE), 3-40
- TEKID (IDENTIFY TERMINAL), 3-40, 2-12
- TEKKPAM (KEYPAD APPLICATION MODE), 3-41, 3-14
- TEKKPNM (KEYPAD NUMERIC MODE), 3-41, 3-14
- TEKOM (ORIGIN MODE), 3-34, 3-42
- TEKORM (OVERSTRIKE/REPLACE MODE), 3-34, 3-42
- TEKRC (RESTORE CURSOR), 3-43, 3-10
- TEKSC (SAVE CURSOR), 3-43, 3-10
- TEKSCNM (SCREEN MODE), 3-34, 3-43
- TEKSTBM (SET TOP AND BOTTOM MARGINS), 3-44, 3-4, 3-6
- TEKSWL (SINGLE WIDTH LINE), 3-44
- Tektronix-private commands
 - TEKANM (ANSI-TO-VT52 MODE), 3-34, 3-38
 - TEKARM (AUTOREPEAT MODE), 3-34, 3-38
 - TEKAWM (AUTOWRAP MODE), 3-34, 3-38
 - TEKCKM (CURSOR KEYS MODE), 3-34, 3-39
 - TEKCOLM (COLUMN MODE), 3-34, 3-39
 - TEKDHL (DOUBLE HEIGHT LINE), 3-40
 - TEKDWL (DOUBLE WIDTH LINE), 3-40
 - TEKID (IDENTIFY TERMINAL), 3-40, 2-12
 - TEKKPAM (KEYPAD APPLICATION MODE), 3-41, 3-14
 - TEKKPNM (KEYPAD NUMERIC MODE), 3-41, 3-14
 - TEKOM (ORIGIN MODE), 3-34, 3-42
 - TEKRC (RESTORE CURSOR), 3-10, 3-43
 - TEKSC (SAVE CURSOR), 3-43, 3-10
 - TEKSCNM (SCREEN MODE), 3-34, 3-43
 - TEKSTBM (SET TOP AND BOTTOM MARGIN), 3-44, 3-4, 3-6
 - TEKSWL (SINGLE WIDTH LINE), 3-44
- terminal architecture, 1-4—1-7
- terminal controls. *See* keys, terminal
- terminal model status, 2-11, 5-39, 5-78
- Terminal Settings Report, 2-11, 5-39, 5-82—5-85
 - special inquiry codes, 5-39, 2-11, 5-82
- terminal settings, saving, 5-42
 - answerback string, 2-10
 - ANSI commands, 3-15
 - copier and printer command settings, 2-14
 - graphics commands, 4-2, 5-10
- terminal space, 4-10
- terminal space units, 4-10
- terminal status, in 4010 Status Report, 2-11, 5-40, 5-86
- terminal type, reporting, 2-12
 - DA (DEVICE ATTRIBUTES), 3-19
 - IDENTIFY, 3-48, 2-12
 - TEKID (IDENTIFY TERMINAL), 3-40, 2-12
- terminator, command (TEK mode), 5-10
- text-display commands, dialog area,
 - CLEAR DIALOG SCROLL, 5-14
 - ENABLE DIALOG AREA, 5-20, 4-4
 - ENTER ALPHA MODE, 5-23, 4-3, 4-17
 - SCS (SELECT CHARACTER SET), 3-7, 3-28, A-1
 - SGR (SELECT GRAPHIC RENDITION), 3-7—3-8, 3-30
 - SI (SHIFT IN), 3-32, 3-7, 3-28, A-1
 - SET DIALOG AREA COLOR MAP, 5-51, 3-7
 - SET DIALOG AREA INDEX, 5-53, 3-7, 4-5
 - SET DIALOG AREA LINES, 5-54, 3-2, 4-4
 - SET DIALOG AREA VISIBILITY, 5-54, 3-2, 4-4
 - SET DIALOG AREA WRITING MODE, 5-55, 3-42
 - SO (SHIFT OUT), 3-36, 3-7, 3-28, A-1

INDEX

T

text-display commands, graphics area, 4-21—4-22

- ENTER ALPHA MODE, 5-23, 4-3, 4-17
- GRAPHIC TEXT, 5-26, 4-21
- SET ALPHATEXT FONT, 5-46, 4-21
- SET GRAPHICS AREA WRITING MODE, 5-60, 4-22
- SET GRAPHTEXT CHARACTER PATH, 5-61, 4-21
- SET GRAPHTEXT ROTATION, 5-62, 4-21
- SET GRAPHTEXT SIZE, 5-63, 4-21
- SET TEXT INDEX, 5-74, 4-22

text-editing modes

- ANSI mode, 3-12
- EDIT mode, 3-12, 4-2, 5-43
- VT52 mode, 3-12

text in dialog area

- graphics applications, 4-3—4-6
- screen-editing applications, 3-7—3-9

text in graphics area, 4-21—4-22

- graphtext, 4-21
- alphatext, 4-22

TEXTRENDITION, 3-30

- transmit baud rate, 2-2, 5-47
- transmit delay, 2-8, 5-58, 5-74
- transmit rate limit, 2-2, 5-75
- transparency, 4-12, 4-3, 5-53

U

- underlining, 3-8, 3-30, 3-34, 3-42, 5-55
- Underscore character (), 3-34, 3-42
- United Kingdom character set, 3-28, A-4
- US (ASCII) character set, 3-28, A-2
- user prompts, how to build, 4-5

V

vector commands, 4-18—4-19

- DRAW, 4-18, 5-19
- ENTER VECTOR MODE, 4-17, 5-24
- MOVE, 4-18, 5-33
- SET 4014 LINE STYLE, 5-77
- SET LINE INDEX, 4-18, 5-66
- SET LINE STYLE, 4-18, 5-66

Vector mode, 4-17, 4-18—4-19, 5-24

- entering, 4-17, 4-19, 5-24, 4-18
- exiting, 4-17, 5-14, 4-18
- for creating panels, 4-23

vectors, 4-18—4-19

- drawing explicitly, 4-18, 4-19, 5-19, 5-33
- drawing implicitly, 4-18, 4-19, 5-24
- undrawn, 5-33

vertical scrolling, 3-4, 3-29, 3-37

VERTICAL TAB (VT), 3-44

VIEWATTRIBUTES, 5-75

viewport, pixel, 5-69

visibility, dialog area, 3-2, 4-4, 5-54

volatile macros, 4-32—4-40

- creating, 5-16
- deleting, 4-40, 5-16

volatile memory, 4-41

VT (VERTICAL TAB), 3-44

VT100 Terminal, emulating 3-2, 1-6

VT52 commands, 3-45—3-48

- conventions, 3-15—3-16
- grouped by function, Section 3 *ANSI/VT52 Commands* divider
- syntax, 3-15—3-16

VT52 mode, 3-14

- command descriptions, 3-45—3-48
- entering, 3-12, 3-29, 3-34, 3-38, 5-43
- syntax for, 3-15—3-16

VT52 Terminal, emulating, 3-2, 1-3, 1-6

W

WINDOW, 5-76

windows, 4-10

wipe index, 5-75

writing modes

- ALU, 5-13, 5-38, 5-41
- in dialog area, 5-55, 3-42
- in graphics area, 5-60, 4-22

X

XMTDELAY, 5-74

XMTLIMIT, 5-75

XON-XOFF. *See* flagging

xy-coordinate parameter

- P_1 characters in, 5-28
- encoding, 5-4—5-5
- encoding subroutine, 6-6
- host, 5-4—5-5
- Setup, 5-6

xy-coordinate report

- decoding subroutine, 6-9
- decoding, 5-80