TEK PROGRAMMERS REFERENCE

Part No. 070-4893-00
Product Group 18

4107/4109 COMPUTER DISPLAY TERMINAL

Please Check for CHANGE INFORMATION at the Rear of This Manual

First Printing NOV 1983

Tektronix®
COMMITTED TO EXCELLENCE

MANUAL REVISION STATUS

PRODUCT: **4107/4109 Computer Display Terminal**

This manual supports the following versions of this product: Firmware Version: 1 and up. Current Version: 2.

REV DATE	DESCRIPTION
NOV 1983	Original Issue

CONTENTS

Section 5 4100-STYLE PARAMETER TYPES, COMMANDS, AND REPORTS

ILLUSTRATIONS

TABLES

Section 1

INTRODUCTION

This manual contains the reference information needed to develop and maintain applications software for the TEKTRONIX 4107 and 4109 Computer Display Terminals.

Related manuals include:

- *The 4107/4109 Computer Display Terminal Operators Manual*, which describes each terminal, its use, and the self-test features.
- *The 4107/4109 Computer Display Terminal Reference Guide*, which contains essential reference material in a condensed form.

WHERE TO LOOK FOR INFORMATION

This manual is organized as follows:

- Section 1, *Introduction*, contains introductory information about the terminal.
- Section 2, *Communications*, discusses the concepts of terminal communications with a host computer and with the terminal's two-port peripheral interface, handshaking protocols, and reports.
- Section 3, *The Graphics Terminal*, discusses basic terminal concepts that you will need to write applications programs for the terminal. Topics include dialog area, basic graphics concepts, graphics input (GIN), and color.
- Section 4, *Screen Editing*, discusses text editing concepts. This section also contains the detailed command descriptions for the ANSI X3.64 and VT52 text editing commands, along with an explanation of the command syntax.
- Section 5, *4100-Style Parameter Types, Commands, and Reports*, contains an alphabetically organized dictionary of all terminal commands except Ansi and VT52 commands, along with an explanation of the command syntax used in this section.
- Appendices include code charts for each character set and their keyboard macro numbers, reference lists of commands, the Tektronix Color Standard, examples of integer parameters, and an index.

FEATURES

The following paragraphs describe the main features of the 4107 and 4109 terminals.

The Display. The terminal incorporates a 60-Hz non-interlaced raster-scan display for flicker-free viewing. The 4107 has a 13-inch display, and the 4109 has a 19-inch display. Each color crt has 640-by-480 pixel resolution.

Alphanumerics. You can display uppercase and lowercase characters. The characters have definable attributes including character color, background character cell color, underline, and blink. The characters are displayed in 7-by-9 dot matrices in a 8-by-15 dot character cell.

Alternate Character Sets. You can select from eight alternate character sets. These sets are: ASCII, United Kingdom, French, Swedish, Danish/Norwegian, German, a special supplementary character set, and a special rulings character set.

Dialog Area. A user-definable dialog area displays host communications without interfering with the graphics on the screen. You can select a column width of 80 or 132 characters, and you can adjust the area height from 2 to 32 lines. The dialog area's scrollable memory is adjustable from two to as many lines as memory permits. The dialog area can be made visible or invisible by host command or with the Dialog key.

Graphics Area. The graphics area is separate from the dialog area. This portion of the display is used for graphics information.

Color. You can select from a palette of 64 distinct color mixtures on the 4107, and from a palette of 4096 distinct color mixtures on the 4109. Up to 16 colors can be displayed simultaneously in the graphics area, with an additional 8 colors in the dialog area. Color can enhance the visual impact and the information content of both graphics and text, and adds a new dimension to text editing.

Graphics. You can draw solid or dashed lines in color. You can fill panels with solid colors, color shading, or with a variety of patterns. You can add labels or text to the graphics in different sizes, colors, and rotation angles. A variety of capabilities provide for versatile graphics programming:

- *Segment operations* allow you to define a group of graphics objects as a *segment* and manipulate this segment as a single entity.
- *Pixel operations* allow the host to access and manipulate individual pixels on the color raster display.
- *Multiple graphics surfaces* can be used for multi-level graphics applications such as architectural or circuit board design.
- *Graphtext commands* allow user- or software-defined fonts for special display purposes. Graphtext can be scaled, rotated, or slanted in a variety of ways for custom-designed graphics displays.

Multiple Views. You can define up to 64 separate views, and use them to window in on different parts of the same display or on different displays. You can display different views simultaneously or in rapid succession.

Zoom and Pan. The zoom and pan features allow you to examine different parts of a picture in detail or get a bird's-eye overall view of a picture.

Graphics Input. An extensive set of graphics input (GIN) commands provide for interactive graphics with the host. You can use most graphics input application programs written for 4010 Series terminals without modification.

Nonvolatile Memory. This portion of the terminal's memory lets you configure the terminal for a particular application and then save the operating parameters in a part of the terminal's memory that is not erased even when the power is turned off. When you turn the power on again, the terminal automatically remembers and uses the saved parameter values.

Commands. The terminal has four types of commands, each designed for a particular purpose. (See *Modes of Operation* later in this section.)

Software Compatibility. The following software packages are compatible with this terminal:

- TEKTRONIX Plot 10 Interactive Graphics Library
- TEKTRONIX Plot 10 Easy Graphing II
- SASGRAPH, from the SAS Institute, Inc
- DISSPLA, from ISSCO (Integrated Software Systems Corporation)
- Certain editing programs designed for use with VT100 or VT52 terminals, such as EDT, VI, and EMACS

You can run most programs written for 4010 Series terminals.

Computer Interface. The terminal uses a full-duplex serial RS-232-C interface with data rates to 38400 baud.

Color Hardcopy. The terminal is compatible with the TEKTRONIX 4695 Color Graphics Copier.

Peripheral Ports. Two RS-232 peripheral ports allow you to connect several compatible Tektronix peripheral devices, including plotters and the 4957 Graphics Tablet for graphics input.

Keyboard. The terminal has a low-profile detachable keyboard. The keys include ASCII uppercase, lowercase, and control characters, BREAK and ERASE keys, a 14-key numeric keypad, and four special function keys and eight programmable function keys. Most keys have N-key rollover. All keys are programmable except Shift, Ctrl, and Caps Lock, and you can set all keys to repeat when held down for more than one-half second.

THE TERMINAL'S PROGRAMMING MODEL

You can configure the terminal to perform many different functions. Once you've set the operating parameters or characteristics of the terminal to best perform the desired function, you can save these parameters in the terminal's memory so that they are not erased when the terminal is turned off. In this way, the terminal can remain in the same configuration for as long as you want.

This terminal is really two different terminals in one package:

- A graphics display terminal that is compatible with Tektronix 4010 Series graphics applications programs. It also has an interactive color interface (used to adjust the terminal's color map or displayed colors), a scrollable dialog area that displays host communications while leaving the graphics display undisturbed, and a subset of the Tektronix 4110 command set.
- 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) 6429 standards. These standards define a set of terminal functions that let you manipulate and edit computer text files. You can configure the terminal to run most popular screen editing programs.

Modes of Operation

The terminal has four major modes of operation. In each mode the terminal accepts one type of *command syntax* that is recognized only in that mode. The modes are as follows:

- **Ansi mode.** In this mode, the terminal understands the syntax of the ANSI X3.64 text editing commands only. One specific implementation of Ansi mode is Edit mode, in which the operating characteristics of the terminal have been set to specific values. The *Screen Editing* section of this manual contains a discussion of Ansi and Edit modes, and the descriptions of the ANSI commands.
- **Tek mode.** In this mode, the terminal understands the syntax of the graphics and terminal control commands. *The Graphics Terminal* and *4100-Style Parameter Types, Commands, and Reports* sections of this manual contain a discussion of this mode and descriptions of the commands available in this mode. For added versatility, the parameters set in Tek mode carry over, as appropriate, into Ansi mode; some parameters set in Ansi mode are also effective in Tek mode.

CAUTION

Many command parameters in Tek mode must be encoded in special formats. See the introductory material in the 4100-Style Parameter Types, Commands and Reports *section of this manual for details.*

- **VT52 mode.** In this mode, the terminal is compatible with programs using VT52-style commands.
- **Setup mode.** In this mode, the terminal accepts Setup mode commands, which the user enters from the terminal keyboard. *The Graphics Terminal* and *4100-Style Parameter Types, Commands, and Reports* sections of this manual contain a discussion of this mode and descriptions of the commands available in this mode.

Figure 1-1 shows these modes and what you can do in each one.

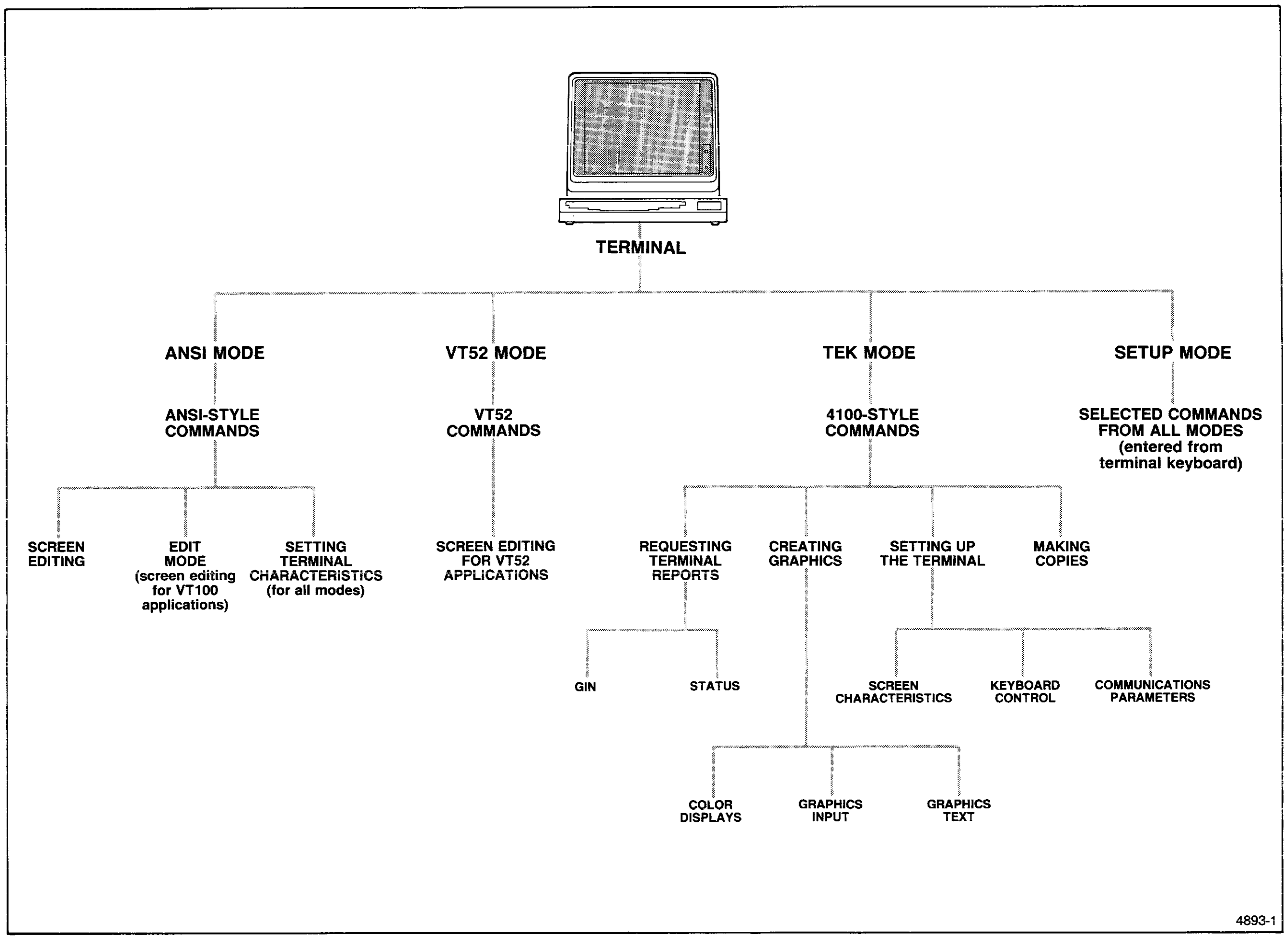


Figure 1-1. Terminal Modes.

Section 2

COMMUNICATIONS

This section discusses basic concepts for communication between the terminal and a host computer or one of the terminal's peripheral ports. The topics covered are:

- *Host Communications Concepts*
- *The Terminal's Input Queue and Handshaking Protocols*
- *Requesting Reports From the Terminal*
- *Port Communications*

HOST COMMUNICATIONS CONCEPTS

This subsection explains basic host communications concepts. You can change the terminal's communications settings by sending commands from the host, or the operator can change them by typing Setup mode commands on the keyboard. All communications parameters can be saved in the terminal's nonvolatile memory by the SAVE NONVOLATILE PARAMETERS command; they are then remembered even if the terminal is turned off.

The following paragraphs explain each communications concept; each discussion concludes by giving the descriptive name and the Setup name of the command used to change the terminal setting.

BAUD RATES

The *baud rate* is the rate in bits per second that the terminal transmits or receives data. For example, 2400 baud means data is transferred at the rate of 2400 bits per second. Communications with the host cannot take place unless the terminal's baud rates are set correctly.

Descriptive name: SET BAUD RATES
Setup name: BAUDRATE

TRANSMIT RATE LIMITS

You can specify a transmit rate limit — a maximum speed for terminal-to-host communications — which may 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, in sending characters to the host, will space those characters so that the average data rate is 300 bits per second. This limit is useful at high terminal-to-host baud rates, where the host computer cannot accept characters at the full data rate specified by the SET BAUD RATES command.

Descriptive name: SET TRANSMIT RATE LIMIT
Setup name: XMTLIMIT

ECHO

Unless the terminal is in Setup or Local mode, the characters that you type on the keyboard are sent to the host. Characters that appear on the screen are really *echoes* of those typed characters. These echoes only appear if the host or modem sends the same characters back to the terminal (called *remote echo*) or the terminal provides its own echo (called *local echo*).

The terminal can be set for either local or remote echo with the SET ECHO command. If you set local echo when the host or modem already provides an echo, the operator will see double characters, LLIIKKEE TTHHIISS. If you set remote echo but the host or modem does not provide an echo, typed characters will not appear on the screen.

Descriptive name: SET ECHO
Setup name: ECHO

FULL-DUPLEX DATA COMMUNICATIONS

The terminal uses full-duplex data communications. To use full-duplex data communications with remote echo, the terminal's local echo must be turned off (enter ECHO NO in Setup mode).

PARITY

The terminal's parity setting controls how the terminal sets the eighth bit (parity bit) in each character it sends to the host. The parity setting must match the kind of parity that the host expects. The SET PARITY command description lists and explains the possible settings.

Descriptive name: SET PARITY
Setup name: PARITY

STOP BITS

While communicating with the host, the terminal sends and receives each character serially, as a sequence of ten or eleven bits. This is called *asynchronous serial* data communications. The first bit for each character is a start bit, always a 0 (or space) bit. The next seven bits determine the particular ASCII character, after which comes a parity bit, as described earlier. The character ends with one or two stop bits, which are always 1 (or mark) bits. The communications line then remains in the marking condition until the start bit for the next character is detected.

While transmitting characters to the host, the terminal includes one or two stop bits in each character it transmits. The terminal ignores the number of stop bits in characters coming from the host.

Descriptive name: SET STOP BITS
Setup name: STOPBITS

BREAK TIME

Pressing the terminal's Break key sends a break signal to the host. In full-duplex communications, the break sets the communication line in a "space" condition for the length of the break. This provides one way to interrupt host communications when software conditions do not allow (for example, when the keyboard is locked).

The factory default value for the break signal is 200 milliseconds; this is adequate for most host computers. (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.

Descriptive name: SET BREAK TIME
Setup name: BREAKTIME

BYPASS MODE AND THE BYPASS-CANCEL CHARACTER

Sometimes you will want the terminal to ignore characters from the host. For example, when the terminal sends a report to the host, if the host echoes the report characters back to the terminal, the terminal could print them as alphatext or interpret them as xy parameters, depending on the terminal 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 leaves Bypass mode and discards the bypass-cancel character.

If the bypass-cancel character is not a NU (Null character, ADE 0), the terminal automatically enters Bypass mode before it sends a report to the host. If the bypass-cancel character is set to the NU character, the terminal cannot enter Bypass mode.

You can also place the terminal in Bypass mode with the ENTER BYPASS MODE command. For example, you might put the terminal in Bypass mode temporarily to suppress the echo of a user's password during a login procedure.

You should set the bypass-cancel character to the last character that the host echoes when the terminal sends a report. For example, if the host echoes CRLF when it receives a CR, set the bypass-cancel character to LF; if the host echoes only a CR, set the bypass-cancel character to CR. If the host does not echo at all, set the bypass-cancel character to NU.

COPING WITH ^{D}T FILLER CHARACTERS

Some host computers intersperse ^{D}T (Delete or Rubout; ADE 127) characters 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.

The terminal includes two features that help solve this problem. First, the terminal 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:

- Write driver routines to send ^{E}c**?** in place of a ^{D}T character. Also change routines that issue integer and xy-coordinate parameters to output ^{E}c**?** instead of ^{D}T.
- Send an IGNORE DELETES command followed by a SAVE NONVOLATILE PARAMETERS command. This sets the terminal to ignore ^{D}T characters: the setting will be remembered, even if the terminal is turned off.

THE TERMINAL'S INPUT QUEUE AND HANDSHAKING PROTOCOLS

Often the terminal cannot process data immediately upon receiving 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, if the operator places the terminal in Local or Setup mode, no information from the host will be processed until the operator cancels the mode. Likewise, there may be times when the host cannot process data as fast as the terminal sends it. Unless some provision is made for controlling communications, data can be lost.

There are several methods for preventing such data loss. Setting the proper baud rates is important; the SET TRANSMIT RATE LIMIT command can slow down data transmission for selected applications, as discussed earlier in this section. There are three other methods for more general control of data flow:

- The terminal has an *input queue* to store data from the host until it can be processed.
- There are two different *handshaking* schemes:
  - Flagging — prevents data overflow at the terminal
  - Prompt mode — prevents data overflow at the host

THE INPUT QUEUE

The terminal's *input queue* (or buffer) is a portion of terminal memory that acts as a "holding tank" for data waiting to be processed. Data from the host or a peripheral port is placed in the input queue; when the terminal is ready for that information it is removed from the queue and processed (commands are executed, graphics displayed, etc.). This frees memory that can then be used to store additional information in the queue.

For example, the terminal cannot display characters while it is erasing the screen. Incoming characters are stored in the input queue until the erase operation is finished, and then the terminal reads the characters from the queue and processes them.

While the terminal is in Setup mode, incoming characters are stored in the input queue. The terminal waits to process those characters until the operator terminates Setup mode.

If the queue fills up, additional incoming data is simply discarded; no more data can be stored in the queue until some old data is removed. A very small queue may overflow frequently, causing important data to be lost. On the other hand, a large queue ties up valuable terminal memory that could be used to display graphics. The best queue size for each application will be determined by experience.

Descriptive name: *SET QUEUE SIZE*
Setup name: *QUEUESIZE*

THE NEED FOR HANDSHAKING

The terminal can display simple alphanumeric text and graphics up to a maximum continuous data rate of 19200 baud. At higher data rates, or for more complex operations, a *handshaking protocol* should be used to prevent the input queue from overflowing. Even at slow data rates, it may be wise to use a handshaking protocol.

Flagging

Flagging prevents the terminal's input queue from overflowing by allowing the terminal to start and stop data transmissions. Two types of flagging are available: DC1/DC3 software flagging and DTR/CTS hardware flagging. The type of flagging you use must match the type used by the host.

DC1/DC3 Flagging. DC1/DC3 flagging uses the DC3 and DC1 control characters ($^{D}1$ and $^{D}3$, with ADEs 17 and 19, respectively) to inhibit or enable the transmitting device. DC3 is the stop character and DC1 is the start character. This allows you to use flagging with devices which do not have control of the RS-232-C DTR and CTS lines. Note that if a DC1 or DC3 character is garbled in transmission, the handshake with the host computer may cause communication to stop.

The terminal can be set to use DC1/DC3 flagging on 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 RTS (Request To Send). If the host is ready to receive the data, it asserts CTS (Clear To Send). The terminal can transmit only when CTS is asserted.

If the terminal transmits characters faster than the host can process them, the host can drop CTS. 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 (Data Terminal Ready) 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. 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 is usually not 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.

Descriptive name: SET FLAGGING MODE
Setup name: FLAGGING

Prompt Mode

Prompt mode uses a character string called a *prompt string*, a waiting interval called a *transmit delay*, and the concept of a *line* of data to manage terminal-to-host transmissions so that the host's input queue does not overflow. It works like this:

1. When the host is ready to receive data, it sends a prompt string to the terminal.
2. When the terminal receives the prompt string, it waits for the transmit delay. During this time, the terminal does nothing.
3. After the transmit delay has elapsed, the terminal sends one line of data to the host.
4. Steps 1, 2, and 3 are repeated until Prompt mode is terminated.

In Prompt mode, a *line* of data consists of all characters waiting to be transmitted up to and including the next EOM (end-of-message) character(s) (there can be one or two of them) or the next EOL (end-of-line) string. The EOL string is typically a CR (Carriage Return).

Descriptive names:	*SET EOL STRING*
	SET EOM CHARACTERS
	SET TRANSMIT DELAY
	SET PROMPT STRING
	PROMPT MODE

Setup names:	*EOLSTRING*
	EOMCHARS
	XMTDELAY
	PROMPTSTRING
	PROMPTMODE

LINES OF TEXT AND THE TRANSMIT DELAY

The preceding discussion mentioned the concepts of line and transmit delay. The following description explains these concepts in more detail.

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 modes) and any report messages that have been requested by the host.

The terminal sends characters in the output queue to the host one line at a time; that is, it sends characters from the output queue to the host until it reaches the end of a line. If there are no characters backed up in the queue, waiting to be sent, then each character is sent as soon as it is typed (even in Prompt mode). If the operator types faster than characters can be sent, then there will be some characters backed up in the queue. In either case, when the terminal encounters a CR it knows that the end of a line has been reached and waits until the *transmit delay* has elapsed before sending any more data to the host. (In Prompt mode, the terminal also waits to receive a prompt string from the host before sending the next line.)

As discussed earlier, a *line of text* means "all the characters waiting to be transmitted, up to and including the next EOM character or EOL string." As the terminal is shipped from the factory, its only EOM character is CR, and the EOL string consists of one character, CR. Thus, if the terminal is set as it is when shipped from the factory, then a *line of text* means "all characters waiting to be transmitted, up to and including the next CR character."

REQUESTING REPORTS FROM THE TERMINAL

An application program can find out various kinds of terminal status information by sending *report commands*. Each such command causes the terminal to send a particular report back to the host. The host's program must be written to interpret these reports. For this reason, the syntax of each report is explained in detail in the *4100-Style Parameter Types, Commands, and Reports* section of this manual.

The terminal ends each line in a report with an EOM character. (The terminal actually substitutes its current EOL string for each EOM character that is sent. The default EOL string is CR, although other characters may be selected with the SET EOL STRING command.)

REPORT COMMANDS

The following paragraphs list the report commands that you can send to the terminal and summarize what information the terminal can return for each command. See particular REPORT commands and REPORT message types for details.

REPORT 4010 STATUS

This command causes the terminal to send a 4010 status report.

- If the terminal is not in GIN mode, the report includes a terminal status byte and the position of the alphanumeric (dialog area) cursor.
- If the terminal is in GIN mode, the report includes the graphic cursor position.

REPORT DEVICE STATUS

This command causes the terminal to send a device status report for one of the terminal's communications ports:

HO:	the Host port
P0:	Peripheral Port 0
P1:	Peripheral Port 1
HC:	the Color hardcopy interface

REPORT ERRORS

This command causes the terminal to generate an errors report message. This message reports the eight most recently detected error codes, their severity levels, and how many times each was detected.

REPORT GIN POINT

This command causes the terminal to generate one GIN report. This report is also sent when the operator causes a GIN event. The contents of this report depend on what GIN function-device code combination is currently enabled. Possibilities are:

- Locator or Stroke report — returns the ASCII key pressed and the xy-location of the GIN cursor.
- Pick report — returns the ASCII key pressed, the cursor position, and information about the segment picked.

REPORT PORT STATUS

This command returns status information (baud rate, parity, flagging mode, etc.) for a specified peripheral port.

REPORT SEGMENT STATUS

This command allows the host to interrogate the terminal for information about one or more graphic segments. Possible information returned includes the segment's position, visibility, display priority, etc.

REPORT SYNTAX MODE

This command causes the terminal to report which syntax mode was selected by the most recent SELECT CODE command.

REPORT TERMINAL SETTINGS

This command allows the host to interrogate the terminal for various types of information: amount of free memory, terminal model number, firmware version number, and current parameter values for a specified command.

Each time one of these reports is generated (except for the 4010 Status Report), the terminal enters Bypass mode. In Bypass mode the terminal will not accept commands or any other input except the bypass-cancel character. The terminal remains in Bypass mode until it receives the bypass-cancel character or until the Cancel key is pressed. Refer to the *4100-Style Parameter Types, Commands, and Reports* section of this manual for more information about the BYPASS CANCEL CHARACTER command, as well as the individual REPORT commands and messages.

PORT COMMUNICATIONS

The terminal's *two-port peripheral interface* allows it to communicate with several peripheral devices. Each device has its own communications protocol, and the terminal includes several commands to configure output to each port. The concepts for port communications are essentially the same as those for host communications. Following is a list of the commands that control port communications.

SET PORT BAUD RATE
Setup name: PBAUD
Sets a single baud rate for all data transfer between the terminal and the specified port.

SET PORT EOF STRING
Setup name: PEOF
Sets the end-of-file string for the specified port.

SET PORT FLAGGING MODE
Setup name: PFLAG
Sets the flagging mode used for the specified port.

SET PORT PARITY
Setup name: PPARITY
Sets the parity for input/output to the specified port. There are some differences between port parity modes and host parity modes.

SET PORT STOP BITS
Setup name: PBITS
Sets the number of stop bits in characters sent to the specified port.

REPORT PORT STATUS
Setup name: none
Sends a status report of parameter settings for the specified port.

MAP INDEX TO PEN
Setup name: PMAP
Assigns color indices to pen numbers so that all colors with a given index are plotted with the same plotter pen.

PORT ASSIGN
Setup name: PASSIGN
Assigns a basic commuications protocol to a given port. A general purpose protocol allows you to transmit any RS-232 data. Other protocols allow graphics from the terminal to be interpreted by the various plotters supported by the terminal.

PLOT
Setup name: PLOT
Plots all visible segments on a plotter attached to the specified port.

COPY
Setup name: COPY
Copies information from the host to the specified port.

Section 3

THE GRAPHICS TERMINAL

INTRODUCTION

ABOUT THIS SECTION

This section explains the graphics features of the terminal and introduces the terminal's commands in Tek mode. Read this section to learn how individual commands work together to control specific features. Once you have learned the general nature of the commands, refer to the section *4100-Style Parameter Types, Commands, and Reports* for details on each command.

The principle topics in this section include:

- *Introduction.* Information about this section, about the Tek mode command set, and about the terminal's display.
- *The Dialog Area.* Displaying text in a dialog area in front of the graphics area.
- *Basic Graphics.* Drawing lines, markers, panels, and text in the graphics area.
- *Segments.* Storing pictures, or parts of pictures, for redisplay, and manipulating those stored picture segments.
- *Windows and Viewports.* Zooming in on a stored picture to examine it in more detail, and displaying different pictures in different parts of the graphic area.
- *Surfaces.* Splitting the graphics area into several writing surfaces, and assigning viewports to locations on these different surfaces.
- *Color Display.* Selecting up to sixteen colors to use in the graphics area, and another eight colors to use in the dialog area.
- *Graphic Input.* Using the joydisk or an accessory graphics tablet as a graphic input device.
- *Macros.* Storing character strings in the terminal to be retrieved later for a variety of uses.
- *Defining Graphtext Characters.* Designing your own stroke precision graphtext characters.
- *Pixel Operations.* Writing directly into the terminal's raster memory pixel buffer.

NOTE

Examples of commands in this section are in a simplified format; an actual command uses either Setup or Host syntax, as shown in the detailed command description. This section gives just the command's descriptive name followed by a **:** *and the example values for the command's parameters.*

ABOUT THE TEK MODE COMMAND SET

The commands described in this section all belong to the Tektronix 4100 Series terminal command set and operate from the host only in Tek mode. Tek mode commands are also referred to as *4100-style commands*; this term includes both the Host version and the Setup version of Tek mode commands.

NOTE

Before issuing any Tek mode (4100-style) commands from the host computer, be sure that the terminal is set to Tek mode with the SELECT CODE command.

To put the terminal in Tek mode, have the host computer send a SELECT CODE command that specifies Tek mode. The complete set of graphics commands is now available to the host.

An operator can use the graphics commands at any time, regardless of whether the terminal is in Tek mode. To do so, the operator puts the terminal in Setup mode and types the Setup versions of these commands.

ABOUT THE DISPLAY

The terminal displays two types of information:

- Host/operator dialog
- Graphics

The *host/operator dialog* consists of messages from the host and the operator's responses. *Graphics* is the screen's pictorial information. Usually, a host program directs the two types of information to separate areas of the screen. The area of the screen where the host/operator dialog appears is the *dialog area.* The area where graphics is displayed is the *graphics area.* Figure 3-1 shows the idea.

THE DIALOG AREA

Think of the dialog area as a writing surface that sits in front of the graphics area. The dialog area can vary in size from two lines (at the bottom of the screen) to 32 lines (occupying the entire screen). As explained later in "Colors and Transparency," the dialog area need not completely cover the graphics area, and you can make the dialog area transparent.

Either a host program or the operator can set the size of the dialog area by specifying the number of dialog lines. The operator uses the DALINES command; a host program uses the SET DIALOG AREA LINES command.

NOTE

Dialog area features that you set while the terminal is in Ansi mode remain in effect when the terminal enters Tek mode.

DIALOG BUFFER

The terminal can save more lines of dialog than what the dialog area shows. The saved lines, which include the lines displayed, are the *dialog buffer.* A host program sets the number of lines saved with the SET DIALOG AREA BUFFER SIZE command; the operator uses the Setup command DABUFFER. The operator or the host computer can scroll the lines of the dialog buffer, moving different parts of the buffer into view (see Figure 3-1). As more lines of dialog accumulate, the dialog buffer fills up, and the terminal discards the oldest line for every new line saved.

ALPHATEXT

Alphatext is the type of text the terminal uses to display messages from the host. (The other type of text, *graphtext*, is used as labels in pictures. It is described under "Basic Graphics," later in this section.)

Alphatext consists of the ordinary characters that the host sends to the terminal and expects the terminal to display as alphanumeric characters. Characters that are part of commands or graphic coordinates to the terminal are not alphatext.

When the terminal is in Tek mode, it displays alphatext on the screen. (In this state the terminal is said to be in *Alpha mode.*) While the dialog area is disabled, alphatext is displayed in the graphics area. It may appear on the screen in one of two sizes, depending on the SET 4014 ALPHATEXT SIZE command.

When the dialog area is enabled and visible, or the terminal is in Ansi or VT52 mode, the terminal displays alphatext in the dialog area.

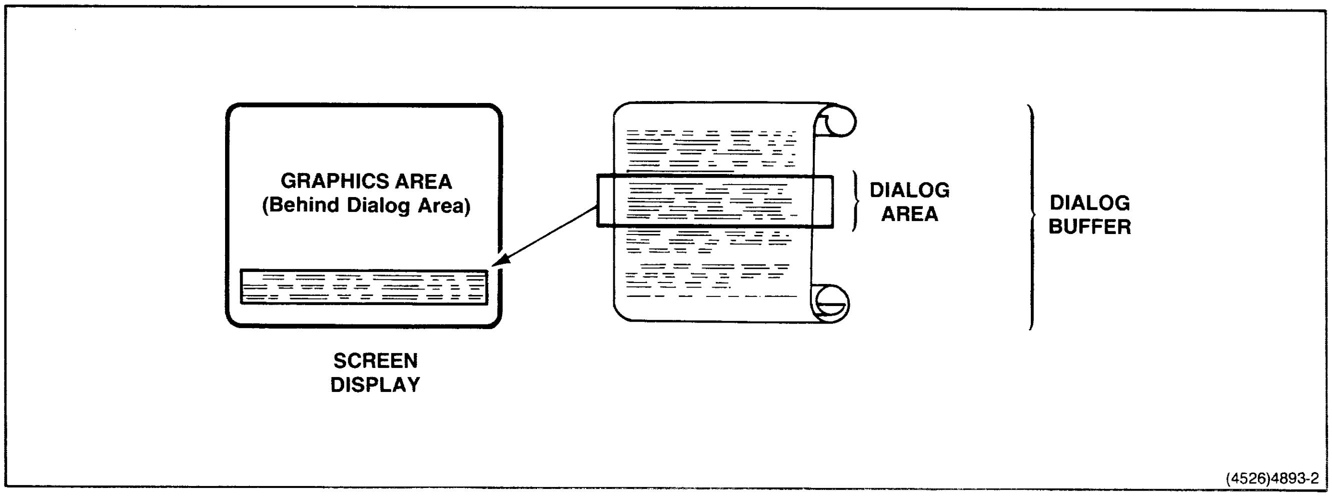


Figure 3-1. Dialog and Graphics Areas.

ENABLING THE DIALOG AREA

When the dialog area is *enabled*, all alphatext is sent to the dialog buffer but not necessarily displayed. A host program enables or disables the dialog area with the ENABLE DIALOG AREA command; the operator uses the Setup command DAENABLE.

When the dialog area is disabled, the terminal emulates a Tektronix 4010 Series terminal. (Such terminals do not have a dialog area.) With the dialog area disabled, alphatext appears in the graphics area at locations determined by previous commands or display operations.

MAKING THE DIALOG AREA VISIBLE

The dialog area does not appear on the screen unless it is *visible*. If the dialog area is enabled, the dialog is saved in the dialog buffer; but dialog is not displayed unless the dialog area is both enabled and visible. If the dialog area is enabled but *invisible*, changing the the dialog area to *visible* displays dialog that accumulated while the dialog area was invisible. A host program sets dialog visibility with SET DIALOG AREA VISIBILITY; the operator uses the Setup command DAVISIBILITY.

COLORS AND TRANSPARENCY

The SET DIALOG AREA INDEX command (Setup command DAINDEX) sets these colors in the dialog area:

- Character color — the color for alphatext in the dialog area
- Character background — the color of the cell surrounding each character
- Dialog background — the color of the dialog area before anything is written on it

Figure 3-2 shows where the dialog area uses these colors.

Both the dialog background and character background can be *transparent*. Graphics behind a transparent area can be seen. For example, when character background is transparent, alphatext appears as if it were written on a piece of glass in front of the graphics.

The details of specifying colors are explained later in this section.

DIALOG AREA COMMANDS

Table 3-1 lists the commands that control the dialog area. Most of these commands can either be sent by the host or be entered by the operator as a Setup command. The SAVE NOVOLATILE PARAMETERS command (Setup mode NVSAVE command) saves the settings made by these commands; these settings are then retained even when the terminal is turned off.

Table 3-1

COMMANDS AFFECTING THE DIALOG AREA

Host Command	Setup Command	Function
ENABLE DIALOG AREA	DAENABLE	Directs alphatext either to the dialog area or graphics area
SET DIALOG AREA BUFFER SIZE	DABUFFER	Sets the maximum number of dialog lines the terminal remembers
SET DIALOG AREA LINES	DALINES	Sets the maximum number of dialog lines visible at one time
SET DIALOG AREA INDEX	DAINDEX	Sets color index of dialog text and background
SET DIALOG AREA COLOR MAP	DACMAP	Defines the colors of the eight dialog area color indices
SET DIALOG AREA WRITING MODE	DAMODE	Sets space and underscore either to replace or to overstrike other characters
SET DIALOG AREA VISIBILITY	DAVISIBILITY	Makes dialog area visible or invisible
CLEAR DIALOG SCROLL	CLEAR-DIALOG or D Eras key	Deletes all characters in the dialog buffer

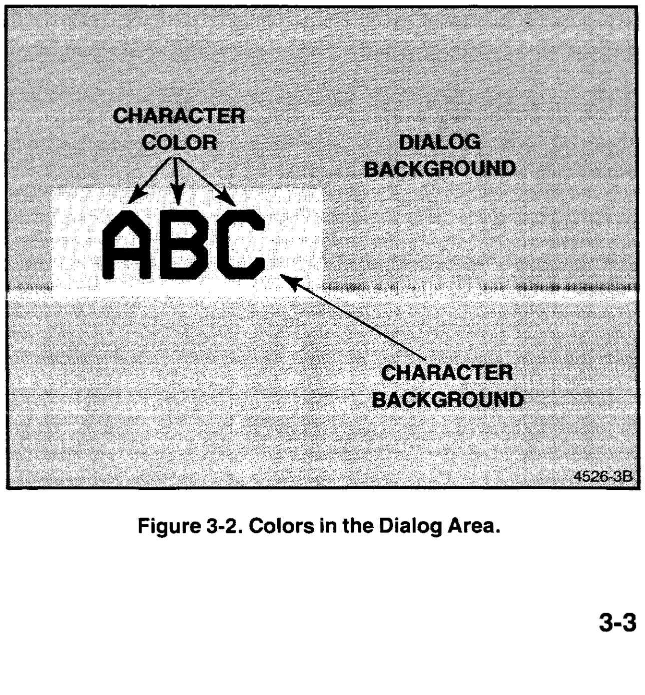


Figure 3-2. Colors in the Dialog Area.

BASIC GRAPHICS

This discussion tells how a program can draw pictures in the graphics area. Subjects covered are:

- *Terminal Space.* How to specify screen positions.
- *Lines.* How to draw lines, and how to make them solid, dashed, or colored.
- *Markers.* How to mark screen positions with small symbols.
- *Panels.* How to fill an area with a pattern or color.
- *Graphtext.* How to put text in your pictures.

TERMINAL SPACE

Graphics are created on a conceptual two-dimensional surface called *terminal space.* All or part of terminal space might be displayed in the graphics area depending on the window you display. (Windows are described later in this section.)

Commands use *terminal space coordinates* to specify locations in terminal space. These are rectangular cartesian coordinates, and when sent to the terminal must be coded a certain way. (See "XY-Coordinates in Host Syntax" in the section *4100-Style Parameter Types, Commands, and Reports* for instructions.)

Figure 3-3 shows the positions certain coordinate pairs indicate. In the figure, the shaded area represents a displayed portion of terminal space. This displayed area (*window*) can be changed with the SET WINDOW command, described later in this section.

You can use any integer from 0 through 4095 for either the x- or the y-coordinate. This range covers the entire terminal space. Each unit in this coordinate system is a *terminal space unit* (TSU).

Current Position

There is always one position in terminal space that is the *graphics beam position*, often called just the *current position.* This is the position that was specified in the last graphics display operation and is usually the default position the terminal uses when none is specified. When the terminal is first turned on, position (0,3071) is the current position. This position is called the *Home* position and is near the upper left corner of the screen when the factory default window is used. (Windows are described later in this section.)

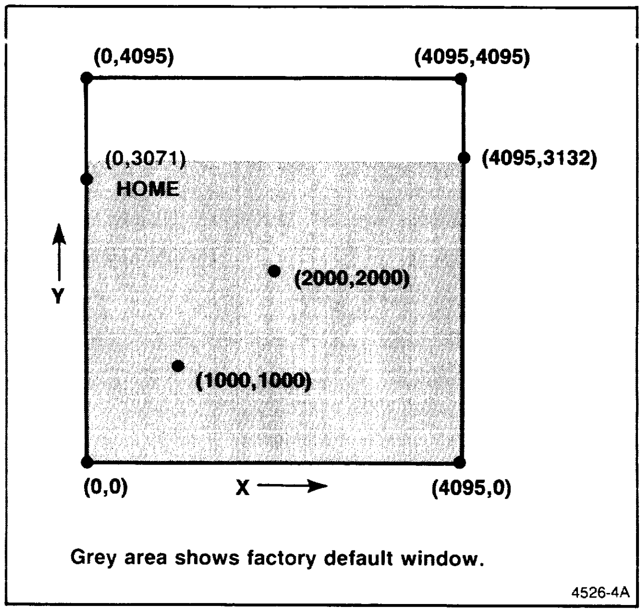


Figure 3-3. Terminal Space Coordinates.

LINES

To draw lines, you can use either of two methods. One method uses a special mode, Vector mode, in which the terminal interprets messages from the host as encoded positions for line segment endpoints. The other method uses MOVE and DRAW commands. Figure 3-4 shows the two methods.

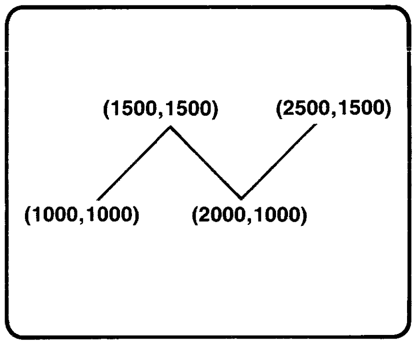
Vector Mode

The ENTER VECTOR MODE command puts the terminal in Vector mode. When in this mode, the terminal interprets all characters the host sends, except for characters forming commands, as xy-coordinates. (See the section *4100-Style Parameter Types, Commands, and Reports* for instructions on sending xy-coordinates.)

The first xy-coordinate pair specifies where the line starts. When the terminal receives a second coordinate pair, it draws a line to that position and makes it the current position. As the terminal receives subsequent coordinates, it repeats the process, drawing a line from the current position to the specified position, and updating the current position. This continues until the terminal leaves Vector mode.

At times, you might want to start a line at a new position without drawing a line to that position. (This is like lifting the pen from the paper in a drawing and moving it to a new location.) To do this, send another ENTER VECTOR MODE command. A new line starts at the next position you specify.

At times, you might want to draw a line from the current position when the terminal is not yet in Vector mode. To do this, issue an ENTER VECTOR MODE command followed by the ${}^{B}{}_{L}$ character. The terminal chimes and draws a line from the current position to the next position you specify.



MOVE and DRAW Method

Command Description	After Encoding
MOVE 1000,1000	${}^{E}{}_{C}$LF'`z'Z
DRAW 1500,1500	${}^{E}{}_{C}$LG+`w+W
DRAW 2000,1000	${}^{E}{}_{C}$LG'`z/T
DRAW 2500,1500	${}^{E}{}_{C}$LG+`w3Q

Vector Mode Method

Command Description	After Encoding
ENTER VECTOR MODE	${}^{G}{}_{S}$
1000,1000	'`z'Z
1500,1500	+`w+W
2000,1000	'`z/T
2500,1500	+`w 3Q
ENTER ALPHA MODE	${}^{U}{}_{S}$

4526-6A

Figure 3-4. Two Methods for Displaying a Line.

MOVEs and DRAWs

The MOVE-and-DRAW method of creating lines uses individual commands to create lines. The terminal does not need to be in Vector mode to execute these commands.

The MOVE command sets the current position, but does not draw a line to that position. The command's effect is analogous to lifting a pen from paper and placing it at a new position in a drawing.

The DRAW command draws a line from the current position to the position specified in the DRAW command. The current position is then changed to the position specified in the command.

Attributes of Lines

Before drawing a line on the screen, you can set both a line style and a line index for the line. The seven available line styles are shown with the detailed description of the SET LINE STYLE command. The line index determines the line's color. With the SET LINE INDEX command you can assign any of the available color indices to the line before drawing it. (Color indices are explained later in this section.)

MARKERS

A *marker* is a small, predefined symbol that marks a position in terminal space. Before displaying a marker, you can choose any of eleven marker types shown in the description of the SET MARKER TYPE command. You then use either of two methods to display markers: one method uses Marker mode; the other method uses an individual command, MARKER, to create each marker.

Marker Mode

The ENTER MARKER MODE command puts the terminal in Marker mode. The terminal then interprets all characters it receives, except for those in commands, as xy-coordinates. It places a marker at each specified position.

The only way to leave Marker mode is to issue the ENTER ALPHA MODE command. Alpha mode is the only mode you can enter directly from Marker mode.

DRAW MARKER Command

The DRAW MARKER command specifies a single position. (In Setup mode, the command name is MARKER.) The terminal displays, at the specified position, the type of marker selected in the most recent SET MARKER TYPE command. The terminal does not have to be in Marker mode when it executes the DRAW MARKER command.

PANELS

A panel is an area in terminal space that can be filled with any of 156 predefined patterns. Figure 3-5 gives some examples of panels.

These steps describe how to create a panel.

1. Send the SELECT FILL PATTERN command and specify the fill pattern you want to use. For the available choices, see the detailed description of this command.
2. Send the BEGIN PANEL BOUNDARY command. This specifies the starting position of the panel's boundary line and whether that boundary line should be displayed in the finished panel.
3. Define the panel's boundary line. Do this in either of two ways:
   a. Put the terminal in Vector or Marker mode and then send the coordinates of boundary segment endpoints.
   b. Send MOVE and DRAW commands.

   It is not necessary to define the last segment which closes the panel.
4. Send the END PANEL command. This automatically creates a boundary line segment back to the starting position and displays the panel with its fill pattern. The current position is set to the panel's starting position.

If the terminal is in Marker mode during this process, markers are not displayed.

If the terminal is in Vector mode, the boundary line is displayed as the panel is created. Depending on the choice specified in the BEGIN PANEL BOUNDARY command, the boundary line is either displayed in the finished panel or covered by the fill pattern.

If a panel's boundary crosses itself as in Figure 3-5C, the terminal uses this rule to determine which areas are inside the panel: If, starting from a point distant from the panel, you cross the boundary an odd number of times to get to the area, the area is in the panel; if you cross the boundary an even number of times, the area is outside the panel. Only areas inside the panel are filled.

As shown in Figure 3-5B, a panel can have more than one boundary. To make such a panel, issue additional BEGIN PANEL BOUNDARY commands before ending the panel. When you issue an additional BEGIN PANEL BOUNDARY command, this closes the boundary being created and starts a new boundary at the specified position. The panel is not filled until the terminal receives the END PANEL command.

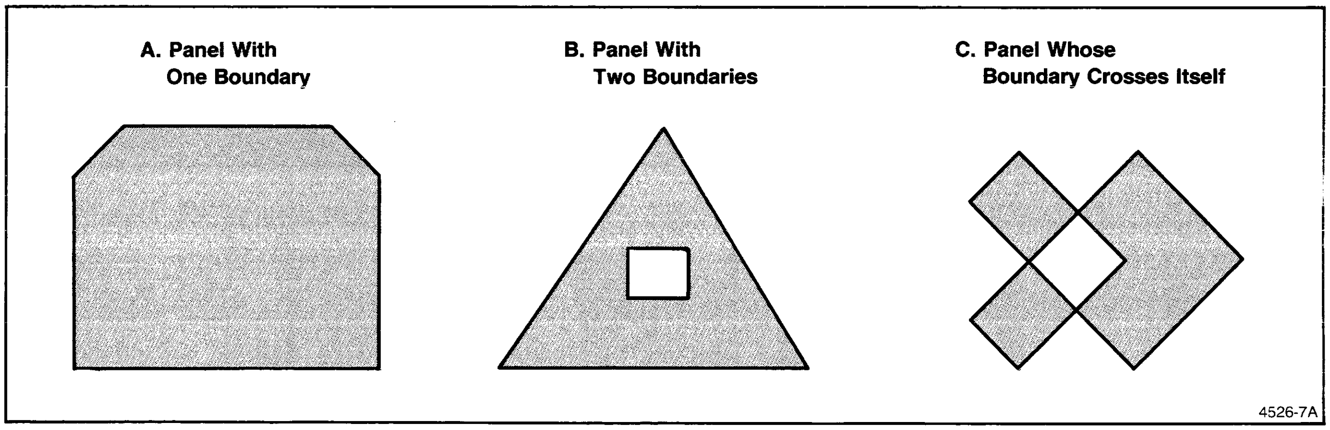

A. Panel With One Boundary
B. Panel With Two Boundaries
C. Panel Whose Boundary Crosses Itself
4526-7A

Figure 3-5. Examples of Panels.

GRAPHTEXT

Graphtext is a special type of text for use in the graphics area. You should always use graphtext in the graphics area because of its versatility and because it leaves the dialog area undisturbed. Graphtext can be resized, rotated, and written in different directions. The terminal lets you put alphatext in the graphics area, but this feature is provided primarily for compatibility with TEKTRONIX 4010-Series terminals.

Before you send the GRAPHIC TEXT command, the command that displays graphtext, you can set several attributes of graphtext with other commands. See Figure 3-6 for a list of these commands and examples.

Only printable characters can be in the string you specify as graphtext. This includes the ASCII characters $^{S}{}_{P}$ through ~ (ASCII decimal equivalents 32 through 126). The terminal detects an error for characters outside this range.

String Precision and Stroke Precision

One of the graphtext attributes is the *graphtext precision*, which you set with the SET GRAPHTEXT PRECISION command. This command selects either *string precision* or *stroke precision*. Stroke precision graphtext is more versatile than string precision graphtext; for example, stroke precision graphtext can be displayed at rotation angles other than 90°.

As shown in Figure 3-6, stroke precision graphtext differs from string precision in several ways:

- String precision graphtext characters can only be rotated in multiples of 90°. For instance, if SET GRAPHTEXT ROTATION specifies an angle of 30°, individual string precision characters remain upright, although the string as a whole will tilt upwards at a 30° angle. Stroke precision characters, however, are each individually rotated at the specified 30° angle.
- When string precision characters are enlarged with the SET GRAPHTEXT SIZE command (or by "zooming in" on a view), they can only be displayed in sizes which are integral multiples of the basic dot matrix character size. In the SET GRAPHTEXT SIZE command, string precision graphtext uses only the height parameter; the width and gap parameters are ignored.
- By contrast, stroke precision text can be displayed with any height, width, or inter-character gap. However, the line segments with which stroke precision characters are drawn do not get any thicker when you make the characters larger.
- The SET GRAPHTEXT SLANT command only works on stroke precision graphtext.
- Different commands are used to switch between character sets for string precision graphtext and stroke precision graphtext.

String precision graphtext is always displayed with the same character set as is used for alphatext. First, a character set is designated as G0 or G1 with the Ansi mode SCS (Select Character Set) command. Then, the current G0 or G1 set is invoked with the Ansi mode $^{S}{}_{I}$ (Shift In) or $^{S}{}_{O}$ (Shift Out) control characters, or with the Tek mode command, SET ALPHATEXT FONT.

Stroke precision graphtext, however, is displayed with the character set chosen by the most recent SET GRAPHTEXT FONT command.

- You can design your own stroke precision fonts, as described later in this section. This is not possible for string precision graphtext.

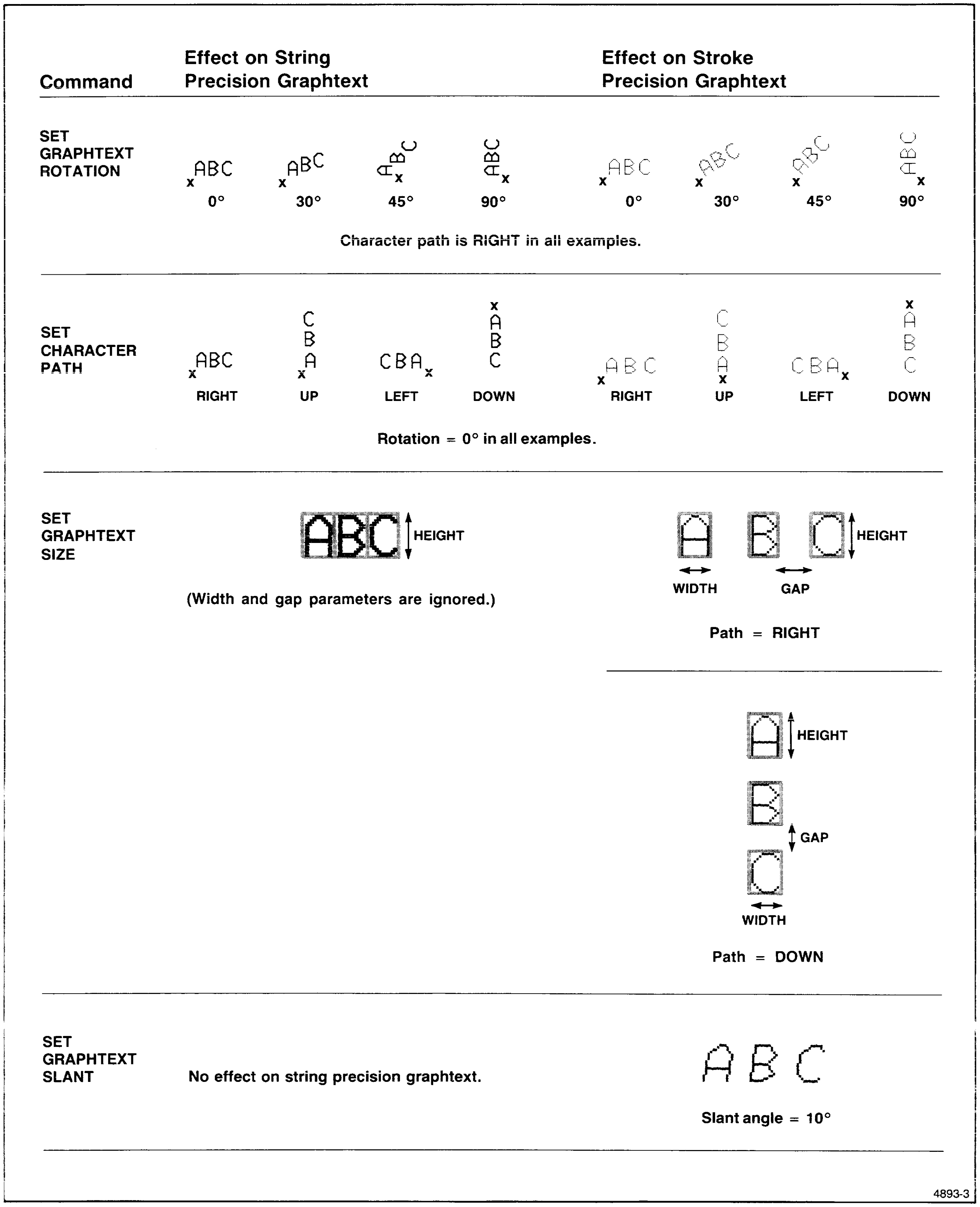

Command
Effect on String Precision Graphtext
Effect on Stroke Precision Graphtext
SET GRAPHTEXT ROTATION
0°
30°
45°
90°
Character path is RIGHT in all examples.
SET CHARACTER PATH
RIGHT
UP
LEFT
DOWN
Rotation = 0° in all examples.
SET GRAPHTEXT SIZE
HEIGHT
(Width and gap parameters are ignored.)
WIDTH
GAP
Path = RIGHT
Path = DOWN
SET GRAPHTEXT SLANT
No effect on string precision graphtext.
Slant angle = 10°
4893-3

Figure 3-6. Commands to Set Graphtext Characteristics.

ALPHATEXT IN GRAPHICS

As mentioned before, alphatext should not be used for labels in pictures. Use graphtext instead.

Alphatext *can* be used in pictures, provided the dialog area is disabled. But this feature is provided only for compatibility with earlier terminals that did not have graphtext. Alphatext cannot be rotated or slanted. The two alphatext sizes (SET 4014 ALPHATEXT SIZE command) have unchangable dimensions. Alphatext does not change size during zooming operations. Consequently, you can't use variable windows, or the terminal's Zoom and Pan keys, to view alphatext in more detail.

SEGMENTS

INTRODUCTION

A *segment* (short for "graphic display list segment") is a graphic object that is defined (saved) in the terminal's internal memory and given a name, or *segment number*. Once you have defined part of a picture as a segment, you can then make it invisible, make it visible again, move it from one part of the screen to another, change its size and orientation angle, and even include it in the definition of another segment.

In order to use some of the terminal's other features, it is necessary first to define graphic information in a segment. For example:

- To let the operator "zoom in" on a picture with the viewing keys, you must first store the picture in the terminal's memory as one or more segments.
- In order to replace the crosshair GIN cursor with a cursor of your own design, the cursor you design must be defined as a segment.
- In order to make a graphic object blink on or off, that object must be defined as a segment.

DEFINING A SEGMENT

To define a segment, do the following:

1. Send a SET PIVOT POINT command. This specifies a point to be used as a reference point for the entire segment. For instance, when you move a segment to a particular location, the terminal moves the whole segment so that the pivot point is at the location specified. When you rotate the segment, it rotates about its pivot point.

2. Send a BEGIN SEGMENT command. This starts the segment definition, and specifies the segment number (the segment's name). The segment's pivot point is the pivot point specified in the most recent SET PIVOT POINT command.

3. Draw a graphic object. Use any of the commands described under "Basic Graphics," earlier in this section.

4. End the segment definition with an END SEGMENT command.

Besides BEGIN SEGMENT and END SEGMENT, you can use some other, shorter, commands:

- BEGIN NEW SEGMENT is a combination of END SEGMENT and BEGIN SEGMENT; it ends the segment currently being defined and starts the definition of a new segment with a specified segment number. The pivot point of the new segment is at the current graphics beam position.
- BEGIN HIGHER SEGMENT is similar, but you need not specify the segment number; it is one greater than the segment number of the previous segment.
- Likewise, BEGIN LOWER SEGMENT terminates a segment definition and begins another segment with the next lower segment number.

NOTE

The commands BEGIN NEW SEGMENT, BEGIN HIGHER SEGMENT, and BEGIN LOWER SEGMENT all create segments whose pivot points are at the current graphics beam position. Use these commands for segments that will remain in the same position. Use the BEGIN SEGMENT command for defining segments that you will move, scale, or rotate with the SET SEGMENT POSITION or SET SEGMENT IMAGE TRANSFORM commands.

MOVING SEGMENTS AROUND

Once you have defined a segment, you can change its size (stretch it in the x- and y-directions), rotate it about its pivot point, and move it to another location in terminal space. The command for this is SET SEGMENT IMAGE TRANSFORM (see the command's detailed description).

If you only want to change a segment's position, and not its size or rotation angle, you can use the SET SEGMENT POSITION command, instead.

USING ONE SEGMENT IN THE DEFINITION OF ANOTHER

While defining one segment, you can change the position of another segment and then send INCLUDE COPY OF SEGMENT to include a copy of that segment in the segment you are defining. You can do this repeatedly. For instance, you might draw an apple tree as follows:

1. Delete any Segment 1 or Segment 2 that may currently exist:
   a. DELETE SEGMENT: 1
   b. DELETE SEGMENT: 2
2. Define Segment 1 to represent an apple:
   a. Issue SET PIVOT POINT for a point which will be the center of an apple.
   b. BEGIN SEGMENT: 1
   c. Draw a panel representing a red apple. Draw the panel boundary so as to include the pivot point inside it. (Use SELECT FILL PATTERN to choose a red color for the apple, and then BEGIN PANEL boundary, MOVEs and DRAWs, and END PANEL to draw the apple.)
   d. END SEGMENT
3. Define Segment 2 to represent the apple tree:
   a. BEGIN SEGMENT: 2
   b. Draw a panel representing the green branches of the tree.
   c. Draw a panel representing the brown trunk of the tree.
   d. Move segment one to a location among the branches of the tree. (Issue a SET SEGMENT POSITION command for segment 1.)
   e. INCLUDE COPY OF SEGMENT: 1
   f. Repeat steps 3d and 3e several times to place apples at different places in the branches.
   g. END SEGMENT

GRAPHICS PRIMITIVES WITHIN SEGMENTS

When you define a segment, you draw it using a number of *graphics primitives*, such as MOVE, DRAW, panels, and graphtext. You can vary the attributes (dashed line style, line color, text color, fill pattern for panels) of these primitives.

However, once the segment is defined, you cannot change the primitives or primitive attributes. The only way to edit a segment is to delete it (with the DELETE SEGMENT command) and define it again.

One special graphic primitive is the *pick identification number*. This is used only during graphic input pick operations; it has no effect on the segment's appearance. However, if a segment is to be "picked" in a graphic input operation, and different parts of the segment are to have different pick i.d. numbers, then you must issue the SET PICK ID command from time to time while defining the segment. For more information, see the SET PICK ID command's detailed description.

ATTRIBUTES OF SEGMENTS

Segments have one fixed attribute and a variety of changeable attributes. The fixed attribute is the pivot point, which cannot be changed after beginning the definition of a segment.

Changeable segment attributes are listed along with other segment commands in Table 3-2. For more information, see the descriptions of the individual commands in the section *4100-Style Parameter Types, Commands, and Reports.*

Table 3-2

COMMANDS PERTAINING TO SEGMENTS

Descriptive Name	Setup Mode Command	Function
SET PIVOT POINT	SGPIVOT	Sets the pivot point for new segments not yet defined.
BEGIN SEGMENT	SGBEGIN	Begins a segment definition.
END SEGMENT	SGEND	Ends a segment definition.
BEGIN NEW SEGMENT	SGNEW	Ends one segment definition and starts another.
BEGIN HIGHER SEGMENT	SGHIGHER	Ends one segment definition and starts another with the next higher segment number.
BEGIN LOWER SEGMENT	SGLOWER	Ends one segment definition and starts another with the next lower segment number.
INCLUDE COPY OF SEGMENT	SGCOPY	Puts copy of defined segment into segment being defined.
SET PICK ID	SGPICKID	While defining a segment, marks part of it with a *pick identification number* for later use when picking the segment during a graphic input operation.
DELETE SEGMENT	SGDELETE	Deletes a defined segment.
RENAME SEGMENT	SGRENAME	Changes a segment's name (segment number).
SET SEGMENT POSITION	SGPOSITION	Moves a segment so that its pivot point is at a specified location.
SET SEGMENT IMAGE TRANSFORM	SGIMAGETRANSFORM	Scales the segment in the x- and y-directions, rotates it about its pivot point, and then moves it to a specified position.
SET SEGMENT VISIBILITY	SGVISIBILITY	Makes a segment visible or invisible.
SET SEGMENT HIGHLIGHTING	SGHIGHLIGHTING	Causes a segment to blink on and off: to alternately be visible and invisible.
SET SEGMENT WRITING MODE	SGWRITINGMODE	Chooses whether a segment is to be drawn in Set mode or XOR mode.
SET SEGMENT DISPLAY PRIORITY	SGPRIORITY	Sets the priority order (front to back order) for displaying (and picking) segments which overlap.
SET SEGMENT DETECTABILITY	SGDETECTABILITY	Specifies whether a segment can be picked with the pick graphic input function.
SET SEGMENT CLASS	SGCLASS	Adds a segment to one set of segment classes and deletes it from another set of segment classes.
SET CURRENT MATCHING CLASS	SGMATCHINGCLASS	Specifies the segment matching class in terms of already-defined segment classes.

Use this version of Table 3-2 instead of the version found on page 3-12.

Table 3-2

COMMANDS PERTAINING TO SEGMENTS

Descriptive Name	Setup Mode Command	Function
SET PIVOT POINT	SGPIVOT	Sets the pivot point for new segments not yet defined.
BEGIN SEGMENT	SGOPEN	Begins a segment definition.
END SEGMENT	SGCLOSE	Ends a segment definition.
BEGIN NEW SEGMENT	SGNEW	Ends one segment definition and starts another.
BEGIN HIGHER SEGMENT	SGUP	Ends one segment definition and starts another with the next higher segment number.
BEGIN LOWER SEGMENT	SGDOWN	Ends one segment definition and starts another with the next lower segment number.
INCLUDE COPY OF SEGMENT	SGINCLUDE	Puts copy of defined segment into segment being defined.
SET PICK ID	SGPICKID	While defining a segment, marks part of it with a *pick identification number* for later use when picking the segment during a graphic input operation.
DELETE SEGMENT	SGDELETE	Deletes a defined segment.
RENAME SEGMENT	SGRENAME	Changes a segment's name (segment number).
SET SEGMENT POSITION	SGPOSITION	Moves a segment so that its pivot point is at a specified location.
SET SEGMENT IMAGE TRANSFORM	SGTRANSFORM	Scales the segment in the x- and y-directions, rotates it about its pivot point, and then moves it to a specified position.
SET SEGMENT VISIBILITY	SGVISIBILITY	Makes a segment visible or invisible.
SET SEGMENT HIGHLIGHTING	SGHIGHLIGHTING	Causes a segment to blink on and off: to alternately be visible and invisible.
SET SEGMENT WRITING MODE	SGMODE	Chooses whether a segment is to be drawn in Set mode or XOR mode.
SET SEGMENT DISPLAY PRIORITY	SGPRIORITY	Sets the priority order (front to back order) for displaying (and picking) segments which overlap.
SET SEGMENT DETECTABILITY	SGDETECT	Specifies whether a segment can be picked with the pick graphic input function.
SET SEGMENT CLASS	SGCLASS	Adds a segment to one set of segment classes and deletes it from another set of segment classes.
SET CURRENT MATCHING CLASS	SGMATCHINGCLASS	Specifies the segment matching class in terms of already-defined segment classes.

SPECIAL SEGMENT NUMBERS

Each individual segment that you define has a number in the range from 1 to 32767. Besides these positive segment numbers, however, there are certain other special segment numbers, as follows:

- **Segment 0.** Segment 0 refers to the crosshair graphics cursor. For instance, you can issue a SET SEGMENT POSITION command for Segment 0, and thereby move the crosshair cursor. If a command cannot be used with the crosshair cursor, then specifying Segment 0 causes the terminal to detect an error. For instance, you cannot rotate the crosshair cursor, so the SET SEGMENT IMAGE TRANSFORM command does not accept 0 as a valid segment number.
- **Segment –1.** Segment –1 means "all positive-numbered segments." By specifying –1 in a command to set a segment attribute, you can set that attribute for all the currently defined segments. For instance, you can issue a SET SEGMENT VISIBILITY command for Segment –1 to make all segments invisible, or to make all segments visible again.
- **Segment –2.** Segment –2 means "default for new segments not yet defined." When you issue a command to set a segment attribute for Segment –2, you cause all future segments that you later define to assume the specified attribute. For instance, you can issue a SET SEGMENT VISIBILITY command for Segment –2, and thereby cause all future segments to be visible (or invisible).
- **Segment –3.** Segment –3 means "all segments in the current segment matching class." When you issue a command to set a segment attribute for Segment –3, you cause all segments in the segment matching class to assume the specified attribute. (The segment matching class is described under "Segment Classes," the next topic in this section.)

SEGMENT CLASSES

NOTE

The following description gets rather mathematical. On a first reading, you may want to skip this part and proceed to "Windows and Viewports," the next major topic in this section.

Introduction

A *segment class* is a collection of segments. In mathematical terms, it would be called a *set of segments.* By means of segment classes, and in particular, the *segment matching class,* you can manipulate whole groups of segments together. You can cause whole groups of segments to blink, to change position, or to become visible or invisible.

SET SEGMENT CLASS Command

You place segments in segment classes, or remove them from segment classes, with the SET SEGMENT CLASS command. (The Setup mode name for this command is SGCLASS.) This command has three parameters:

- The first parameter specifies a particular segment which is to be added to, or removed from, one or more segment classes.
- The second parameter is a list of class numbers specifying classes from which the specified segment is to be removed. The host sends this parameter as an array of integers. In setup mode, the operator types a left angle bracket, then several class numbers, and then a right angle bracket.
- The third parameter is a list of class numbers specifying classes to which the specified segment is to be added. It is specified in the same way as the second parameter.

For instance, while becoming acquainted with segment classes, you might try typing the following command in Setup mode:

```
SGCLASS 2 <1  2  3> <4  5  6>
```

This would first remove Segment 2 from Segment classes 1, 2, and 3, and then add Segment 2 to classes 4, 5, and 6.

For the first parameter, you can use the *special segment numbers* just described. Also, in the second and third parameters, you can use "-1" to mean "all segment classes." For instance,

```
SGCLASS -2 <-1> <1  2  3>
```

causes "segments not yet defined" (Segment -2) to be removed from "all segment classes" (class -1), and then added to segment classes 1, 2, and 3. Any segments that you subsequently define would then belong to segment classes 1, 2, and 3, and to no other segment classes.

Set Notation

NOTE

In the following text, the ∩ symbol represents the set intersection operator, the ∪ symbol represents the set union operator, and the overline symbol represents the set complement operator. Thus,

(class 1) ∩ (class 2)

means the intersection of segment classes 1 and 2, that is, the set of all segments which are in segment class 1 and also in segment class 2. Again,

(class 1) ∪ (class 2)

means the union of segment classes 1 and 2, that is, the set of all segments which are either in segment class 1, or in segment class 2, or both. Finally,

$\overline{\textit{(class 1)}}$

means the complement of segment class 1, that is, the set of all segments which are **not** *in segment class 1.*

The Segment Matching Class

The *segment matching class* is a set of segments that is defined in terms of other segment classes. To define the segment matching class, issue a SET SEGMENT MATCHING CLASS command. (In Setup mode, this is SGMATCHINGCLASS.) This command has two parameters:

- The first parameter is a list of class numbers for classes to which segments in the matching class must belong.
- The second parameter is a list of class numbers for classes to which segments in the matching class must NOT belong.

For instance, suppose you want the segment matching class to include all segments that are in class 1 and also in class 2, but *not* in class 3. That is, you want the segment matching class to be the set intersection of class 1 and class 2 with the complement of class 3:

$$\text{(segment matching class)} = \text{(class 1)} \cap \text{(class 2)} \cap \overline{\text{(class 3)}}$$

To accomplish that, you could type the following Setup mode command:

```
SGMATCHINGCLASS <1  2> <3>
```

This sets the segment matching class to the set of segments which is the intersection of segment class 1 with segment class 2 and with the complement of segment class 3.

Using the Segment Matching Class

Once you have placed segments in various classes, and defined the segment matching class in terms of those classes, you can use the special segment number, –3, to mean "all segments in the current matching class."

For instance, if you want to highlight all segments that are in class 1 and class 2, but not in class 3, you could proceed as follows:

```
SGMATCHINGCLASS <1  2> <3>
SGHIGHLIGHTING -3, 1
```

The first command sets the segment matching class to be:

$$\text{(class 1)} \cap \text{(class 2)} \cap \overline{\text{(class 3)}}$$

The second command sets segment highlighting. Its first parameter specifies "all segments in the current matching class" (Segment –3), while its second parameter (1) specifies that highlighting (blinking) is to be enabled for those segments.

More complex combinations of segment classes can always be decomposed into a standard "union of intersections" form. For example, consider this expression:

$$[\text{(class 1)} \cup \text{(class 2)}] \cap [\text{(class 3)} \cup \overline{\text{(class 4)}}]$$

This can be rearranged by the rules of Boolean algebra, as follows:

$$[\text{(class 1)} \cap \text{(class 3)}] \cup [\text{(class 1)} \cap \overline{\text{(class 4)}}] \cup [\text{(class 2)} \cap \text{(class 3)}] \cup [\text{(class 2)} \cap \overline{\text{(class 4)}}]$$

Thus, to highlight all segments in

$$[\text{(class 1)} \cup \text{(class 2)}] \cap [\text{(class 3)} \cup \overline{\text{(class 4)}}]$$

you could type the following Setup mode commands:

```
SGMATCHINGCLASS <1  3>
SGHIGHLIGHTING -3, 1
```

Highlights all segments in $\text{(class 1)} \cap \text{(class 3)}$.

```
SGMATCHINGCLASS <1> <4>
SGHIGHLIGHTING -3, 1
```

Highlights all segments in $\text{(class 1)} \cap \overline{\text{(class 4)}}$.

```
SGMATCHINGCLASS <2  3>
SGHIGHLIGHTING -3, 1
```

Highlights all segments in $\text{(class 2)} \cap \text{(class 3)}$.

```
SGMATCHINGCLASS <2> <4>
SGHIGHLIGHTING -3, 1
```

Highlights all segments in $\text{(class 2)} \cap \overline{\text{(class 4)}}$.

Of course, a host application program wouldn't use the Setup mode commands; it would use the Tek mode SET SEGMENT MATCHING CLASS and SET SEGMENT HIGHLIGHTING commands, instead. See the section *4100-Style Parameter Types, Commands, and Reports* for detailed descriptions of these commands.

WINDOWS AND VIEWPORTS

WINDOWS

A *window* is a rectangular area in terminal space, and is the part of terminal space that you want to view in the graphics area of the screen.

SET WINDOW Command

To specify the size and location of the window, you send a SET WINDOW command. The command includes xy-coordinate parameters for the lower-left and upper-right corners of the window. These positions in terminal space are displayed at the corresponding corners of the current viewport. (The "current viewport" is the part of the screen where the window contents are displayed. The default viewport is the entire screen; with the default viewport, the lower-left corner of the window maps onto the lower-left corner of the screen, and the upper-right corner of the window maps onto the upper-right corner of the screen.)

The display is linearly distorted unless the window has the same aspect ratio as the viewport. (Aspect ratio is the ratio of height to width.) Figure 3-7 shows different windows displaying the same picture in terminal space. The figure assumes that the viewport is the default viewport, which occupies the entire screen.

Zoom and Pan

The operator can place the terminal in *Viewing Keys mode* and use the Zoom, Pan, and View keys to change the size and location of the window in terminal space. With the Zoom key, the joydisk, and the View key, the operator can "zoom in" to examine a part of terminal space in more detail. (That is, the operator makes the window smaller, so that a smaller part of terminal space is displayed on the screen.) With the same keys, the operator can "zoom out" to see more of the picture in terminal space. (He makes the window larger.) The operator can move the window around in terminal space with the Pan key, the joydisk, and the View key. All this is described in the Operators Manual.

For this to work, however, the picture being displayed must be defined in the terminal's memory as one or more graphic segments. Otherwise, pressing the View key causes the terminal to erase its viewport but not to redraw the view. (If the terminal doesn't remember the picture, it cannot draw it again. For the terminal to remember the picture, it must be defined as one or more segments.)

If you don't want the operator to use the Zoom and Pan keys, you can disable them with the LOCK VIEWING KEYS command.

Your applications program can produce a zoom or pan effect simply by redefining the window with the SET WINDOW command. Again, for this to work, the picture being displayed must be defined as one or more segments in the terminal's memory.

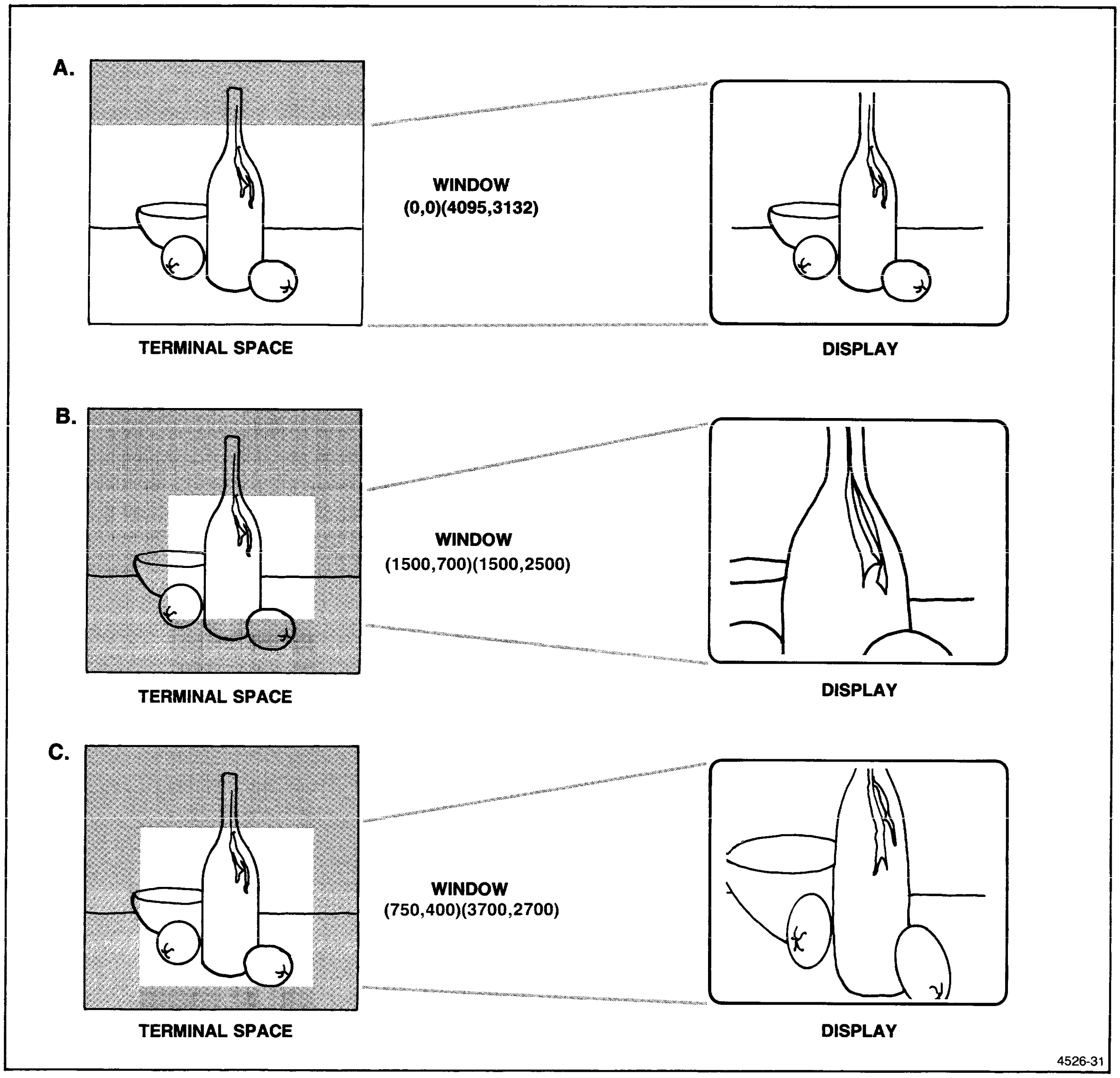


Figure 3-7. Examples of Windows.

VIEWPORTS AND MULTIPLE VIEWS

SET VIEWPORT Command

A *viewport* is a rectangular region on the screen where the contents of the corresponding window are displayed. On power-up, the terminal's viewport occupies the entire screen. You can, however, specify a smaller viewport. To do this, send a SET VIEWPORT command, specifying the lower-left and upper-right corners of the desired viewport. For this command, x-coordinates range from 0 to 4095, and y-coordinates from 0 to 3071. Thus you can get the default viewport by sending the command:

```
SET VIEWPORT: (0,0), (4095,3071)
```

Since the terminal displays the dialog area in front of the graphics area, all or part of the viewport can be obscured by opaque parts of the dialog area. As explained earlier in this section, you can avoid obscuring the viewport by issuing commands that make the dialog area invisible or transparent.

Another way to avoid obscuring the viewport would be to place the viewport and the dialog area in different parts of the screen. Figure 3-8 shows the technique. The SET DIALOG LINES: 8 command makes the dialog area occupy the bottom fourth of the screen. The SET VIEWPORT command causes graphic information to appear in a viewport above the dialog area. The SET BORDER VISIBILITY command draws a border around the viewport.

Creating New Views

You can create several different viewports, each in its own part of the screen. Each viewport is associated with a corresponding window in terminal space, a list of segments visible in that viewport, and a *view number*.

In this manual, the word *view* refers to a collection of several things: a viewport, a window, and a list of visible segments. A viewport determines where a view appears on the screen. A window is in "terminal space" — the coordinate system used for drawing the picture — and determines what part of a picture is displayed.

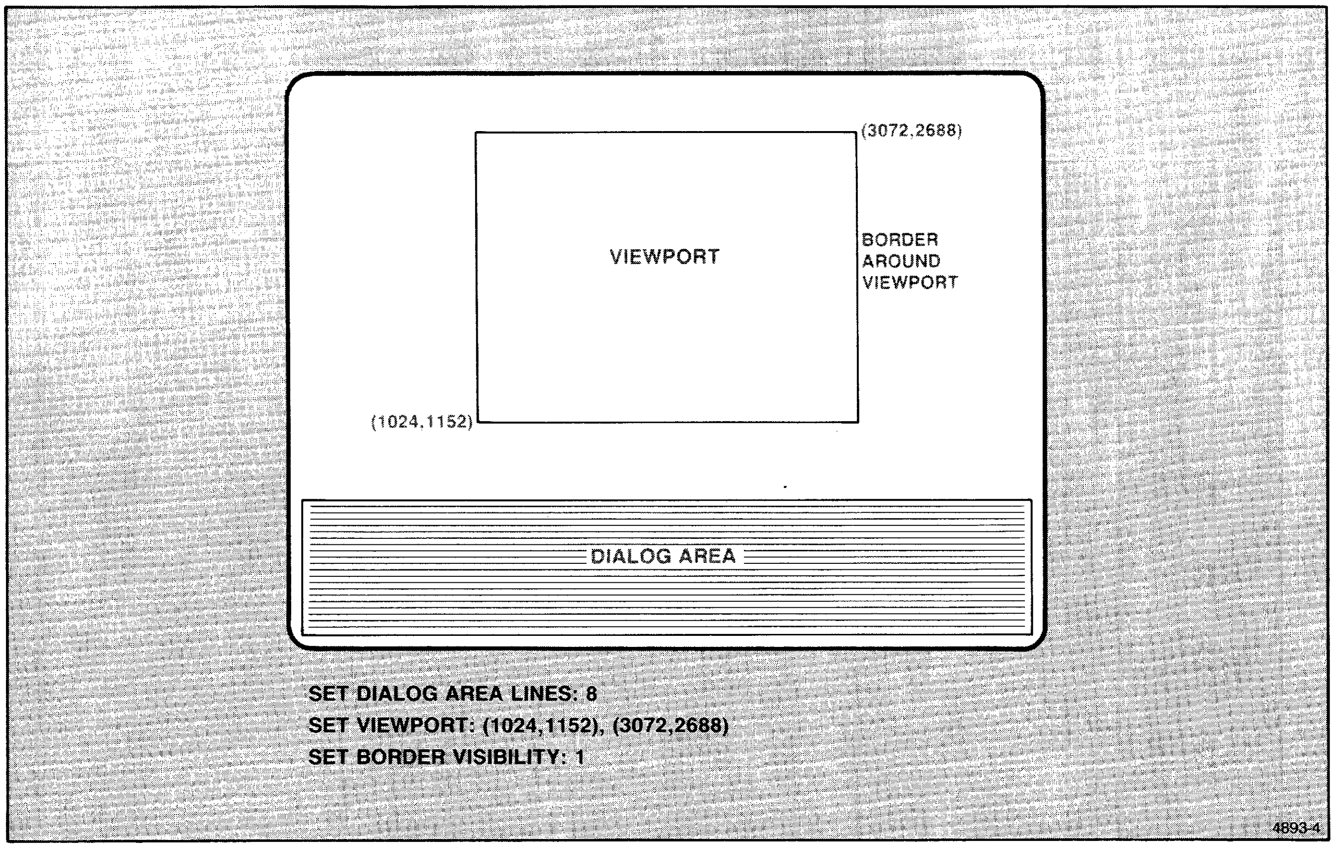


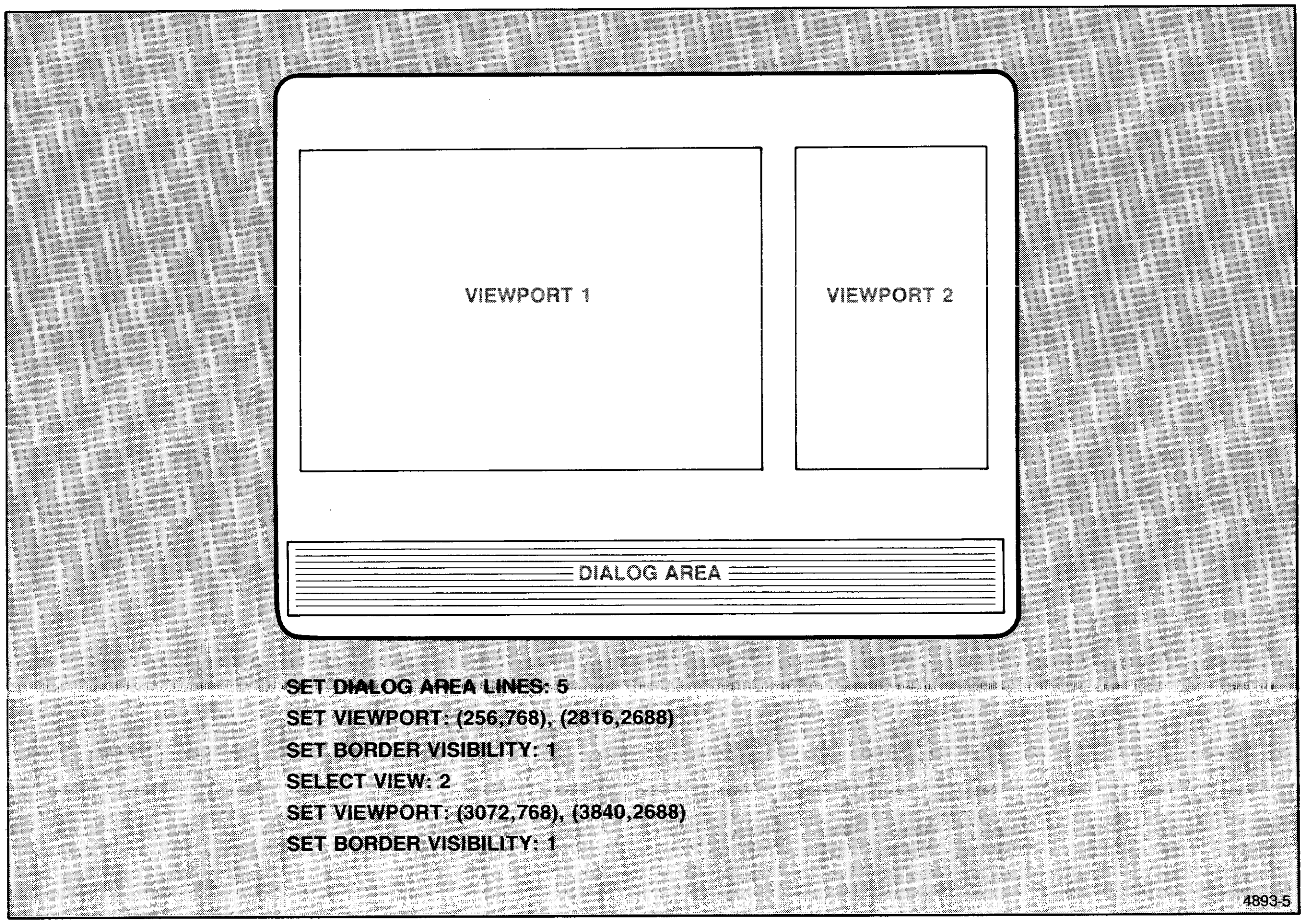
Figure 3-8. Positioning a Viewport Out of the Way of the Dialog Area.

Views are numbered from 1 to 64. On power-up, only one view exists: View 1, with a default viewport occupying the entire screen, a default window extending from (0,0) to (4095, 3136). On power-up, View 1 has no visible segments, because no segments have been defined.

To create several different views, each with its own viewport in a different part of the screen, proceed as follows:

1. Delete all views with a DELETE VIEW: -1 command, and erase the screen. After all views are deleted, the default view, View 1, is automatically created, with the default window and viewport.
2. Send a SET VIEWPORT command to change View 1's viewport so that it does not occupy the entire screen.
3. Create a new view, View 2, and assign a different viewport to it.
   a. Send a SELECT VIEW: 2 command. Since View 2 does not exist, the SELECT VIEW command creates a new view and names it View 2, which becomes the *current view*. Initially, the viewport and window for View 2 are the same as for View 1.
   b. Send a SET VIEWPORT command, specifying the lower-left and upper-right corners for View 2's viewport.
4. Draw any picture that you want displayed in View 2's viewport. If you want the picture to be retained even when zooming and panning, be sure to define the picture as part of one or more segments. Since View 2 is the current view, these segments are visible in View 2, but not in View 1.
5. If you want more than two views, repeat Steps 2 and 3, but with a different view number in the SELECT VIEW command. View numbers can range from 1 to 64, so you can define up to 64 different views.

Figure 3-9 shows how you might organize the screen with two viewports for graphics and a dialog area that does not obscure either viewport.



```
SET DIALOG AREA LINES: 5
SET VIEWPORT: (256,768), (2816,2688)
SET BORDER VISIBILITY: 1
SELECT VIEW: 2
SET VIEWPORT: (3072,768), (3840,2688)
SET BORDER VISIBILITY: 1
```

Figure 3-9. Creating a Screen With Two Viewports.

SET VIEW ATTRIBUTES Command

The SET VIEW ATTRIBUTES command lets you specify three attributes of a view's viewport:

- The *surface* on which the viewport is located. (Graphic display surfaces are explained later in this section.)
- The *wipe index* used for erasing the viewport. This is the color index to which pixels in the viewport are set whenever the viewport is erased. (Color indices are explained later in this section.)
- The *border index*. This is the color index used to draw the border around the viewport.

The Setup mode name for this command is VIEWATTRIBUTES. For more information, see the detailed description of the SET VIEW ATTRIBUTES command.

View Display Clusters

Sometimes it is convenient for several views always to share the same window coordinates in terminal space. (If you zoom in on one view, you zoom in on all.)

You can accomplish this by grouping the views together in a *view display cluster*. See the SET VIEW DISPLAY CLUSTER command's detailed description.

SURFACES

INTRODUCTION

To understand surfaces (and color indices), it helps to know a few things about the terminal's design.

The terminal has several areas of memory, for storing different kinds of information. The *general-purpose memory* holds a variety of things. For instance, graphic segments are stored in general purpose memory as lists of graphic display commands. The *Dialog area buffer* holds only dialog area text. The *nonvolatile memory* stores settings that the terminal must remember even when power is turned off. There is a *color map memory*. (More about that later.) Finally, there is the *raster display memory*, which relates to surfaces.

The terminal's raster memory is used for displaying images in the graphics area. The raster memory has one four-bit word for each *pixel* on the screen. Pixels (picture elements) are the individual colored dots which together compose the image on the screen.

Refer now to Figure 3-10. Think of raster memory as four *bit planes*, each containing one bit (binary digit, 0 or 1) for each pixel in the graphics area of the screen. For each pixel in the graphics area, there are four bits — one from each bit plane — which together compose a binary numeral from 0000 (decimal 0) to 1111 (decimal 15). This binary numeral is the pixel's *color index*: it names one of sixteen colors to be displayed for that pixel.

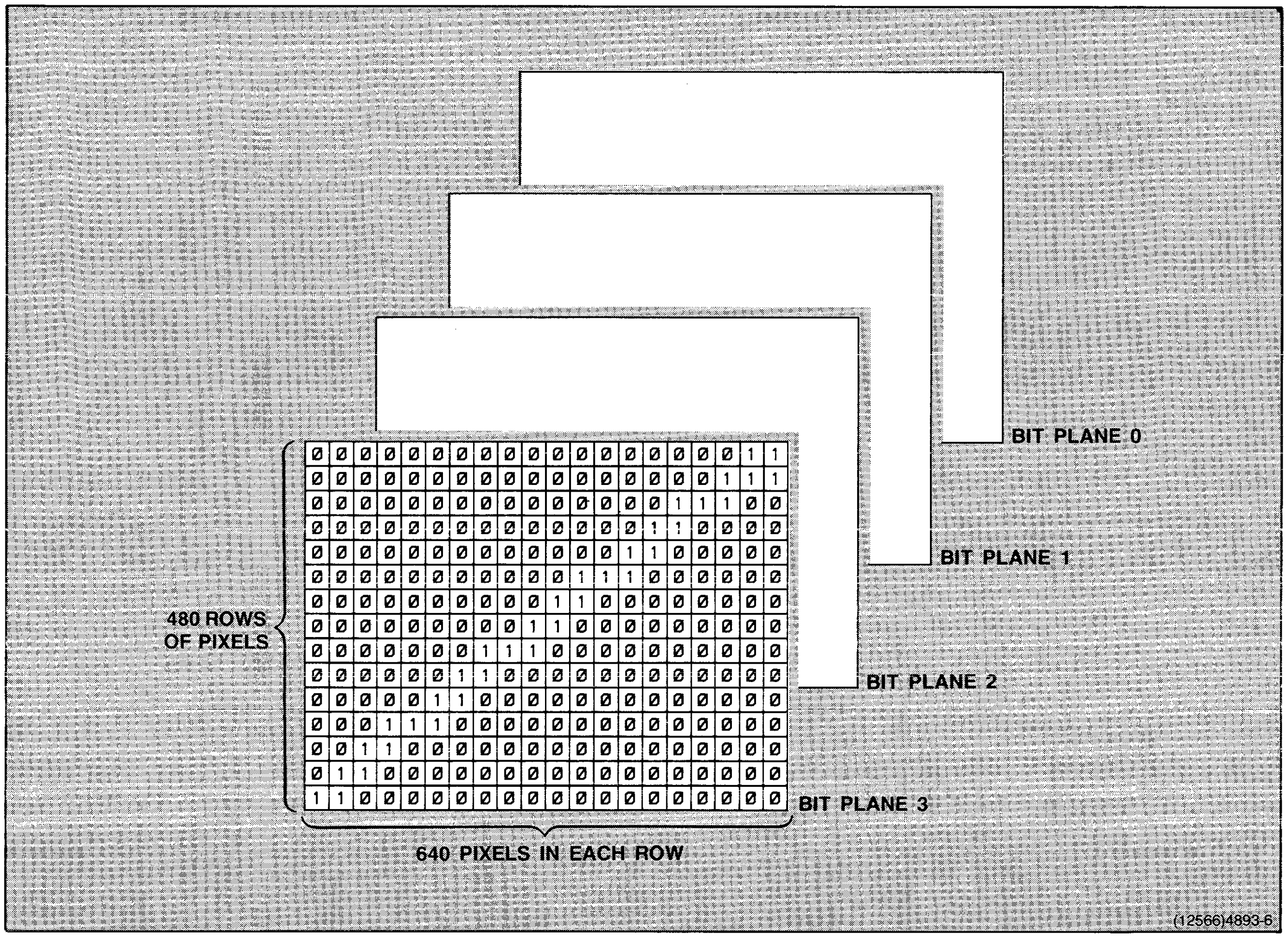


Figure 3-10. Bit Planes in the Terminal's Raster Memory.

Table 3-3 shows the default assignments of colors to color mixtures. The operator can change these assignments with the interactive color interface, described in the Operators Manual. The host computer can change these color assignments with the SET SURFACE COLOR MAP command, described later in this section.

The terminal's internal software draws lines on the screen by changing the color indices stored for each pixel in raster memory. Circuitry in the terminal then automatically reads raster memory and causes the corresponding colors to be displayed on the screen.

Table 3-3

DEFAULT COLOR MIXTURE ASSIGNMENTS

Color Index	Color Mixture
0	Transparent (Black)
1	White
2	Red
3	Green
4	Blue
5	Cyan (Blue-Green)
6	Magenta
7	Yellow
8	Orange (Red-Yellow)
9	Green-Yellow
10	Green-Cyan
11	Blue-Cyan
12	Blue-Magenta
13	Red-Magenta
14	Dark Gray
15	Light Gray

CREATING MULTIPLE GRAPHICS SURFACES

On power-up, the situation is as in Figure 3-1; there is only one graphics surface on which you can draw. However, the SET SURFACE DEFINITIONS command lets you split the terminal's raster memory among several different surfaces, allocating some of the bit planes to each surface. Consider, for instance, the following command:

```
SET SURFACE DEFINITIONS: 2, 2
```

This assigns two raster memory bit planes to one surface, and two bit planes to another surface. Surface 1 is in front, with Surface 2 behind it, and the solid-color background behind Surface 2. If the dialog area is visible, it is in front of all the graphics surfaces. Figure 3-11 shows this in terms of transparent plastic overlays, while Figure 3-12 shows the same thing in terms of the raster memory bit planes.

Initially, there is only one graphics surface (Surface 1), and all four bit planes are assigned to that surface. Therefore, you can draw images using color indices from 0 to 15 (binary 0000 to binary 1111). However, when you split the graphics area into two surfaces, and assign two bit planes to each surface, then each surface can only hold color indices in the range from 0 to 3 (binary 00 to binary 11).

When you split the graphics area into more than one surface with a SET SURFACE DEFINITIONS command, the available color indices on each of the two surfaces assume the default color mixtures. Thus, on Surface 1, color index 0 means transparent, color index 1 means white, color index 2 means red, and color index 3 means green. Likewise for Surface 2.

However, once having defined two surfaces, you can assign different color mixtures to the indices on each surface. (Issue a SET SURFACE COLOR MAP command to set Surface 1's color mixtures. Then issue another SET SURFACE COLOR MAP command to set the color mixtures for Surface 2.) The SET SURFACE COLOR MAP is described later in this section, and in the section *4100-Style Parameter Types, Commands, and Reports.*

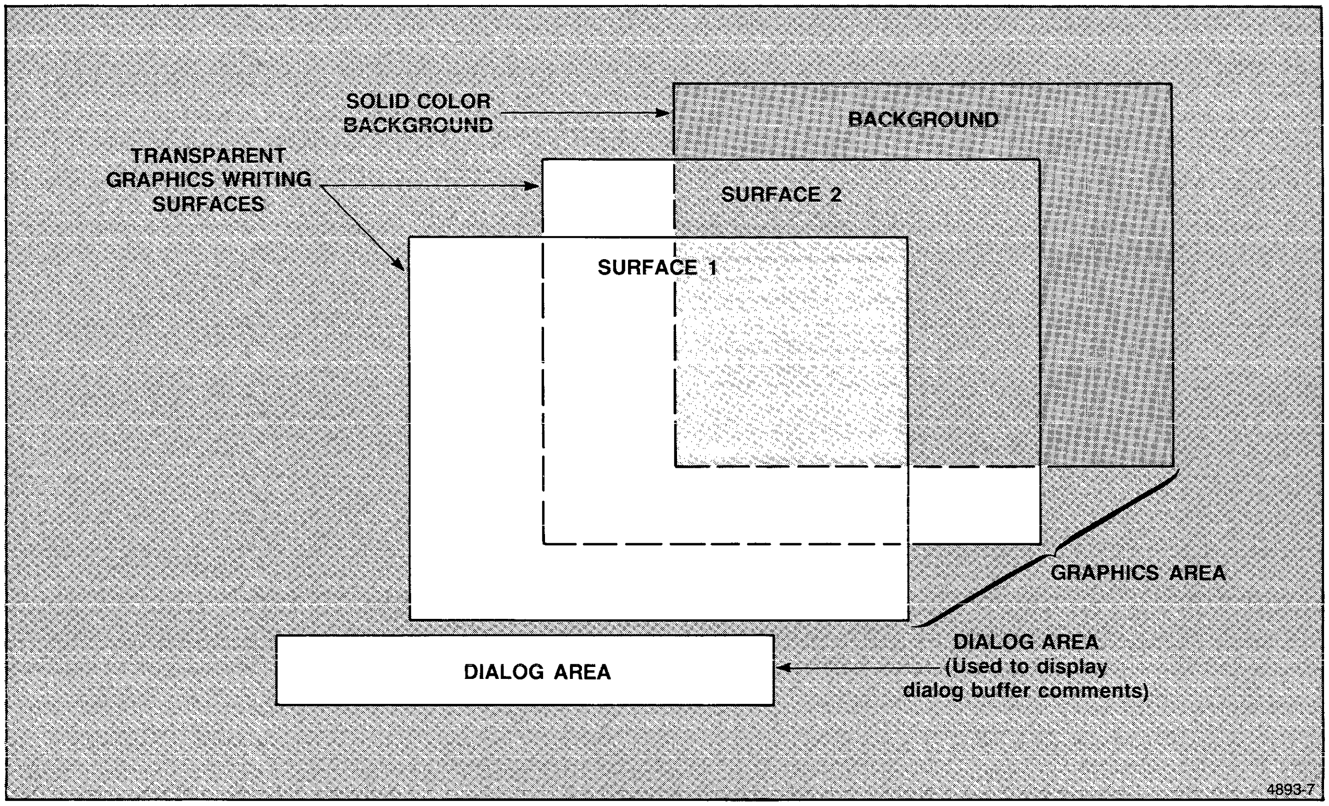

SOLID COLOR BACKGROUND
TRANSPARENT GRAPHICS WRITING SURFACES
BACKGROUND
SURFACE 2
SURFACE 1
GRAPHICS AREA
DIALOG AREA
DIALOG AREA (Used to display dialog buffer comments)
4893-7

Figure 3-11. Splitting the Graphics Area Into Two Surfaces.

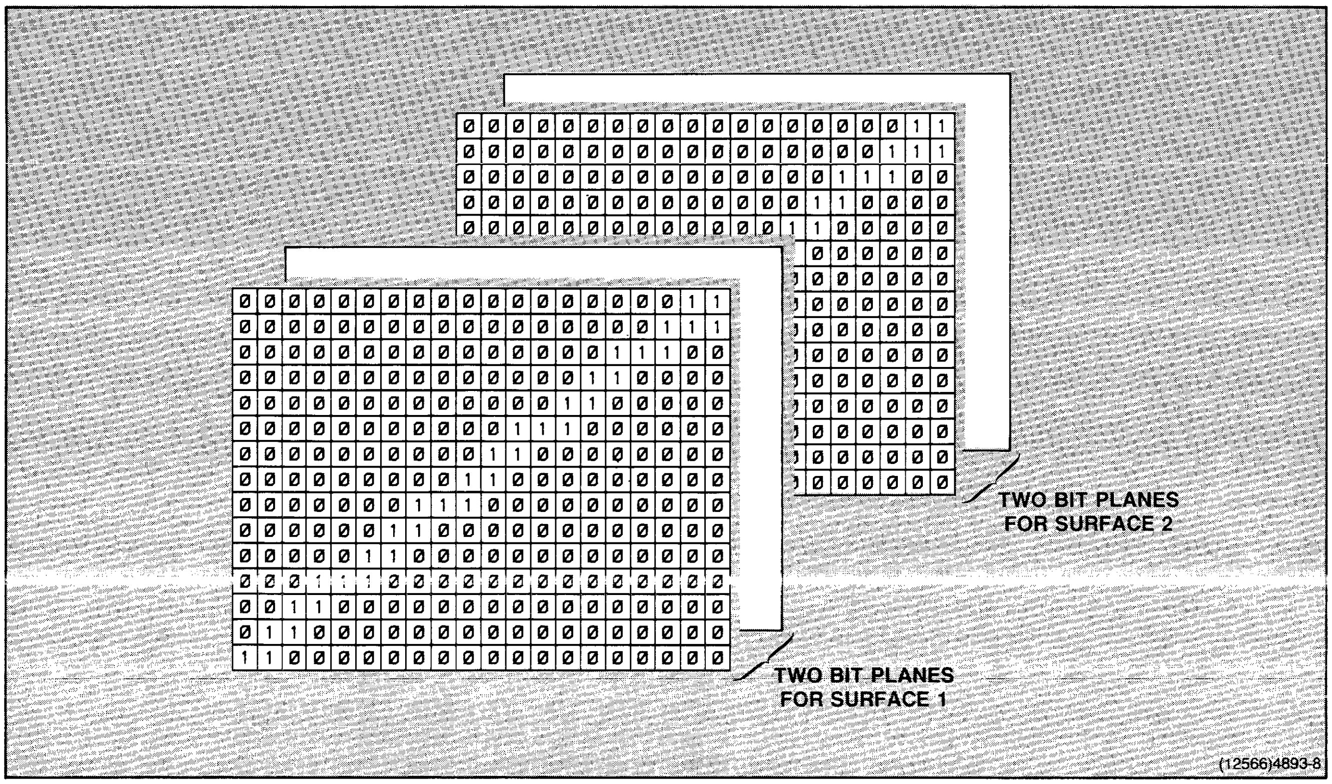


Figure 3-12. Allocating the Bit Planes Between Two Surfaces.

ASSIGNING VIEWPORTS TO DIFFERENT SURFACES

Once you have sent a SET SURFACE DEFINITIONS command to split the graphics area into more than one surface, you can define multiple views as described earlier in this section, and place the viewports for these views on different surfaces.

Consider, for example, the following commands:

SET SURFACE DEFINITIONS: 1, 1, 2
: Creates three surfaces in the graphics area. Surface 1 has one bit plane, so you can only draw on it with Color Index 0 and Color Index 1. Likewise for Surface 2. Surface 3 has two bit planes, so you can draw on it with Color Indices 0, 1, 2, and 3.

SELECT VIEW: 1
: Make View 1 the current view.

SET VIEW ATTRIBUTES: 1, 0, 1
: Place the viewport for View 1 on Surface 1, set the wipe index for View 1 to Index 0 (transparent) and the border index to 1 (white).

SET VIEWPORT: (512,512),(2560,2560)
: Place View 1's viewport with lower-left corner at (512,512) and upper-right corner at (2560,2560). The viewport remains on Surface 1.

SELECT VIEW: 2
: Create View 2 and make it the current view.

SET VIEW ATTRIBUTES: 2, 0, 1
: Place the viewport for View 2 on Surface 2, set its wipe index to 0 (transparent) and its border index to 1 (white).

SET VIEWPORT: (2816,512),(3584,2560)
: Place the Viewport for View 2 with lower-left corner at (2816,512) and upper-right corner at (3584,2560).

SELECT VIEW: 3
: Create View 3 and make it the current view.

SET VIEW ATTRIBUTES: 3, 3, 1
: Place the viewport for View 3 on Surface 3, set its wipe index to 3 (green) and its border index to 1 (white).

Figure 3-13 shows the results of these commands. Note that Viewport 3 has the same lower-left and upper-right coordinates as Viewport 2, although it is on a different surface. When you create a new view with the SELECT VIEW command, the viewport for that view has the same boundaries as the viewport for the previous view — until you change them with the SET VIEWPORT command.

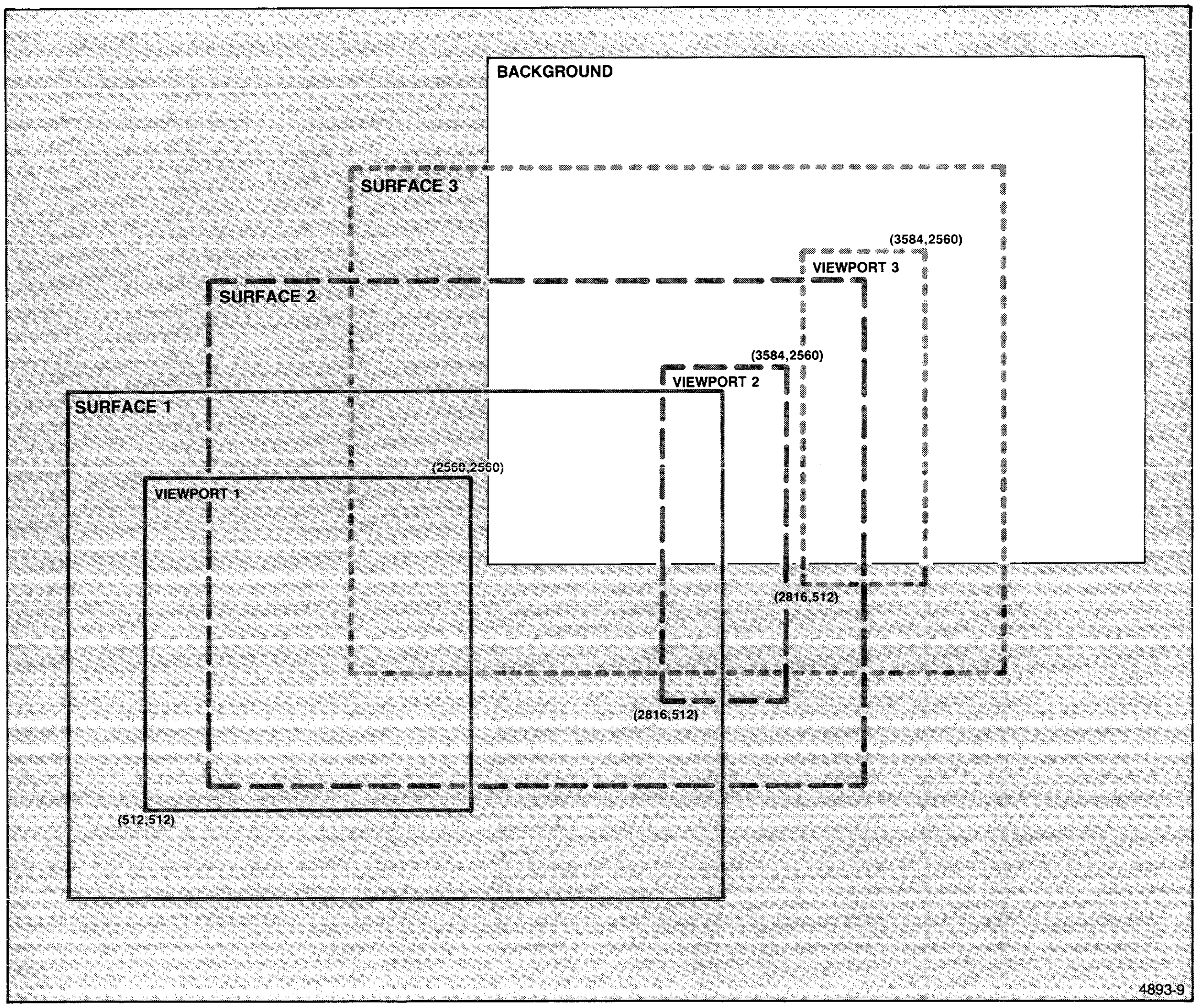


Figure 3-13. Placing Viewports on Different Surfaces.

COLOR DISPLAY

When a command specifies the color of something on the screen, it does not state an absolute color, such as light green or bluish-gray. Instead, it specifies an integer called a *color index*. Either a host program or the operator can select the color associated with a particular color index. For example, when the terminal is shipped from the factory, Color Index 2 is set to mean red. So, anything you assign to Index 2 is displayed in red. If you then change Index 2's definition to blue, anything displayed in Index 2 is blue.

COLOR INDICES

A color index must be an integer in the range from 0 to 15. Actually, the maximum value is 2^n-1, where n is the number of bit planes associated with the surface on which you are drawing. For instance, on power-up, there is only one surface, and that surface has four bit planes; since 2^4-1 is 15, you can draw on that surface with color indices in the range from 0 to 15.

However, suppose you define two graphics surfaces with a SET SURFACE DEFINITIONS command: Surface 1, with one bit plane, and Surface 2, with three bit planes. On Surface 1, 2^1-1 is 1, so you can draw on Surface 1 with only color indices 0 and 1. Surface 2 has 3 bit planes, and 2^3-1 is 7, so you can draw on Surface 2 with color indices in the range from 0 to 7.

The dialog area has its own set of eight color indices.

The color that a color index produces can be set in several ways. When the terminal is turned on, the color indices are assigned default colors. A host program can assign colors with the SET SURFACE COLOR MAP and SET DIALOG AREA COLOR MAP commands. The operator can assign colors with the color interface, which is described in the *Operators Manual*. The operator can also assign colors in the graphics area with the Setup mode command, CMAP, and in the dialog area with the Setup command, DACMAP.

COLOR COORDINATES

Colors for color indices are defined in terms of three *color coordinates*. You can use any of three systems of color coordinates: *HLS*, *RGB*, or *CMY*.

- **HLS (Hue, Lightness, Saturation).** This is the easiest color coordinate system to understand; it is based on a double-ended *color cone*, which is illustrated in the appendices. The **hue** coordinate runs from 0° to 360°, and specifies whether the color mixture is a shade of red, of magenta, of blue, of cyan, of green, or of yellow, etc. The **lightness** coordinate runs from 0% to 100% — from black, through shades of gray, to white. The **saturation** coordinate runs from 0% to 100% and specifies how vivid the color is; 0% would represent a shade of gray, while 100% represents a vivid, fully saturated hue.
- **RGB (Red, Green, Blue).** The RGB color coordinate system represents colors as mixtures of red, green, and blue light. For instance, 100% red, 0% green, and 0% blue yields red, while 100% red, 100% green, and 100% blue yields white.
- **CMY (Cyan, Magenta, Yellow).** The CMY color coordinate system approximates the mixing of cyan, magenta, and yellow inks to produce color mixtures. For instance, cyan and yellow ink mix to produce a green color, so CMY color coordinates of 100% cyan, 0% magenta, and 100% yellow would yield a green color mixture.

To select a color coordinate system, issue a SET COLOR MODE command or the Setup mode CMODE command. (The operator's color interface, however, always uses the HLS system.)

DEFINING COLOR MIXTURES

Once you have selected a color coordinate system with the SET COLOR MODE command, you can define a color mixtures. Remember, however, that Color Index 0 means transparent. (The exception is when index 0 is used as the character color in the dialog area. In that case, index 0, like other color indices, means a particular color mixture.)

To define color mixtures on surfaces in the graphics area, use the SET SURFACE COLOR MAP command or the Setup mode CMAP command. To define the color mixture for the background which lies behind all the transparent writing surfaces, use the SET BACKGROUND COLOR command or the Setup mode CBACKGROUND command.

To define color mixtures in the dialog area, use the SET DIALOG AREA COLOR MAP command, or the Setup mode DACMAP command.

These commands are described in detail in the section *4100-Style Parameter Types, Commands, and Reports.*

BLINKING COLORS

One way to make graphics (or text displayed in the graphics area) to blink on and off is to draw it using a color index that you reserve for blinking objects, and then define the color mixture for that color index to be a *blinking* color mixture. You do this by adding 1000 to the third color coordinate in the SET SURFACE COLOR MAP command.

For instance, suppose you have drawn a picture which includes some graphic text. Reserve a special color index for the text which you want to be able to blink on and off: say, color-index 15. When you want to make the text blink, issue a SET SURFACE COLOR MAP command in which you define index 15 to be, say, "blinking red," as follows:

SET SURFACE COLOR MAP: 1, 15, 120, 50, 1100

In the HLS color coordinate system, this would mean, "on Surface 1, set Color Index 15 to mean HUE 120° (red), LIGHTNESS 50%, and SATURATION 1100 (100% saturation, plus 1000 to make it blink).

When you want to turn the blinking off, you could issue another command to set the color index for index 15 to be, say, "nonblinking white," as follows:

SET SURFACE COLOR MAP: 1, 15, 0, 100, 1000

This would mean, "on Surface 1, set Color Index 15 to mean HUE 0°, LIGHTNESS 100% (white), SATURATION 0 (0% saturation, no blinking).

(There are, however, other ways to make objects blink. You can blink a graphic segment with the SET SEGMENT HIGHLIGHTING command. You can blink an entire graphics surface with the SET SURFACE VISIBILITY command. In the dialog area, you can set a "blink" attribute for subsequent characters with the ANSI mode SGR command.)

EFFECTIVE COLOR DISPLAYS

Color can be used effectively in a number of ways. Some examples are:

- Separating categories of information.
- Helping the operator locate a particular object in a complex display.
- Cuing the operator that information has changed.
- Making small symbols (such as markers) more obvious.

Keep these points in mind when assigning colors:

- The basic set of colors (red, yellow, green, cyan, blue, and magenta) can be readily distinguished one from another and recognized by name. When adding intermediate colors, take care that all the colors are distinguishable.
- Blue text, symbols, and lines often appear fuzzy. It is best to avoid fully saturated blue (HLS = 0, 50, 100) for text, symbols, and thin lines.
- Use the same color the same way in different pictures; this makes it easier to recognize the color's meaning quickly.
- Maintain both a lightness and color difference between characters and their backgrounds. Yellow text on a dark gray background (or the reverse) is highly visible, whereas yellow on a cyan background is not.
- Highly saturated colors often evoke a false depth perception in which some colors stand out from the surface while others recede. For complex displays, it is wise to desaturate the colors somewhat.

GRAPHICS INPUT

Graphics input (GIN) tells a program where the graphics cursor is when the operator presses a key. It also tells which key is pressed.

ENABLING FOR GRAPHICS INPUT

The terminal has two commands which enable it for graphics input:

- The ENABLE 4010 GIN command does simple graphics input in the same way as Tektronix 4010 Series terminals and the Tektronix 4105 terminal.
- The ENABLE GIN command provides more features, and more flexible graphics input. This command lets you specify a different GIN device than the joydisk, such as an accessory graphics tablet. It lets you enable the terminal for more than than one input point per command. Besides the "locator" GIN function, it lets you specify a "pick" function for selecting segments and a "stroke" function for inputting many points from a graphics tablet. You can even enable several GIN devices at once, by issuing several ENABLE GIN commands.

TYPICAL GRAPHICS INPUT SEQUENCES

These steps describe graphics input with the ENABLE 4010 GIN command:

1. The host issues the ENABLE 4010 GIN command. This puts the terminal in GIN mode, and causes it to display the graphics cursor.

2. The operator uses the joydisk to move the graphics cursor. When the cursor is at the desired position, the operator presses one of the ASCII keys.

3. The terminal then sends a 4010 GIN REPORT to the host computer. This report tells the host which key was pressed and the location of the graphics cursor. On sending the report, the terminal exits GIN mode and ceases to display the graphics cursor.

The host must send a separate ENABLE 4010 GIN command for each point.

While the terminal is in GIN mode, certain keys are special:

- If the dialog area is enabled, the terminal ignores the PAGE command and the $^{C}{}_{R}$ character.
- If the dialog area is disabled, the G Eras key, PAGE command, or $^{C}{}_{R}$ character end GIN mode and put the terminal in Alpha mode; no report is sent to the host.

These steps describe typical graphics input with the ENABLE GIN command:

1. The host sends a SET REPORT SIG CHARS command. This specifies the the "signature characters" which the terminal will insert into the GIN report messages to (a) tell the host which of several possible GIN devices caused a particular report message to be sent, and (b) notify the host when the last of several GIN report messages has been sent.
2. The host may send other commands to set other GIN parameters. These parameters are listed in Table 3-4.
3. The host sends an ENABLE GIN command. This puts the terminal in GIN mode, specifies which GIN device is to be used for entering graphicscoordinates, which of three GIN functions (locator, pick, or stroke) is to be performed, and how many coordinates the terminal is to send to the host before exiting GIN mode.

   For instance, the host might specify, "Enable GIN for 5000 'locator' events using the joydisk."
4. The terminal is now in GIN mode, and displays the graphics cursor. This may be the crosshair cursor, or it may be another cursor chosen by the host in Step 2.
5. The operator uses the joydisk (or whatever GIN device was specified in the ENABLE GIN command) to move the graphics cursor around the screen.
6. When the cursor is where the operator wants it to be, the operator signals a *GIN event*. The operator does this by pressing a key (if using the joydisk) or pressing a tablet stylus or tablet puck button (if using an accessory graphics tablet).
7. The terminal sends a *GIN report* to the host computer. This is a GIN LOCATOR REPORT, GIN PICK REPORT, or GIN STROKE REPORT, according to whether the GIN device was enabled for the *locator*, *pick*, or *stroke* graphics input function.
8. Steps 5 and 6 are repeated until either (a) the number of GIN events specified in the ENABLE GIN command have occurred, or (b) the operator presses the CANCEL key to remove the terminal from GIN mode.
9. The terminal sends one last report to the host, telling it that it has exited GIN mode. (This is the FINAL GIN REPORT ITEM included with the GIN REPORT SEQUENCE detailed description.)

LOCATOR, PICK, AND STROKE FUNCTIONS

There are three graphics input functions: *locator*, *pick*, and *stroke*. All three functions are available with the ENABLE GIN command. The ENABLE 4010 GIN command, however, supports only the locator function.

With the *locator* function, the terminal reports the location of the cursor and which key the operator pressed.

The *pick* function lets the operator pick a segment (graphics object) by pointing to it with the cursor. The terminal reports the cursor position, which key was pressed, which segment was picked, and even which part of the segment was picked. (Parts of a segment can be labled with different "pick identification numbers" by issuing SET PICK ID commands when the segment is defined.)

The *stroke* function is only for use with an accessory graphics input tablet. It lets the operator input a stream of points by stroking the stylus or puck across the tablet surface. The SET GIN STROKE FILTERING command lets you specify how close together the reported points may be, either in space or in time.

GRAPHICS INPUT COMMANDS

Table 3-4 lists the GIN commands. For more details, see the command descriptions in the section *4100-Style Parameter Types, Commands, and Reports.*

GRAPHICS INPUT REPORTS

When the terminal is enabled for GIN with the ENABLE 4010 GIN command, it sends its GIN report message back to the host in the 4010 GIN REPORT format. For more information, see the 4010 GIN REPORT detailed description.

When the terminal is enabled for GIN with the ENABLE GIN command, it sends its report messages back to the terminal in the GIN REPORT SEQUENCE format. If it was enabled for the locator function, the GIN REPORT SEQUENCE includes zero or more GIN LOCATOR REPORTs. Likewise, if enabled for the pick or stroke function, the GIN REPORT SEQUENCE includes GIN PICK REPORTs or GIN STROKE REPORTs. See the detailed descriptions of GIN REPORT SEQUENCE, GIN LOCATOR REPORT, GIN PICK REPORT, and GIN STROKE REPORT for more information.

While the terminal is sending a report to the host, the terminal is in Bypass mode. While in this mode, the terminal ignores input from the host. After the message is sent, the terminal returns to Alpha mode. (Bypass mode is explained in the *Communications* section.)

Table 3-4

GRAPHICS INPUT COMMANDS

Command Name	Setup Mode Name	Function
ENABLE 4010 GIN	(none)	Enables the terminal for one GIN locator event with the joydisk.
ENABLE GIN	GINENABLE	Enables the terminal for one or more locator, pick, or stroke events with the joydisk or an accessory tablet.
DISABLE GIN	GINDISABLE	Cancels the effect of an ENABLE GIN command.
SET GIN CURSOR COLOR	GCURSOR	Specifies the color mixture for the crosshair cursor.
SET GIN AREA	GINAREA	Specifies which area on a GIN tablet corresponds to a specified window in terminal space. This may be the window for the current view, or it may be the window specified by SET GIN WINDOW.
SET GIN WINDOW	GINWINDOW	Specifies a window in terminal space onto which the GIN area is mapped.
SET PICK APERTURE	GINAPERTURE	Specifies how close the cursor must be to a segment in order to "pick" that segment.
SET GIN GRIDDING	GINGRIDDING	Constrains the GIN cursor to lie only at the intersection points of an invisible grid.
SET GIN INKING	GININKING	Causes the terminal to draw lines from one graphic input point to the next.
SET GIN RUBBERBANDING	GINRUBBERBANDING	Causes a "rubber band line" to be displayed from the last graphic input point to the cursor position.
SET GIN DISPLAY START POINT	GINSTARTPOINT	Specifies a starting point for the first rubberband line and the first inking line.
SET GIN STROKE FILTERING	GINFILTERING	For "stroke" input from a tablet, specifies how close the input points may be in space or in time.

MACROS

INTRODUCTION

A *macro* is a string of characters identified by an integer called the *macro number*. Either the host or the operator can assign a string to a macro number. The host uses the DEFINE MACRO or DEFINE NONVOLATILE MACRO command, while the operator uses the DEFINE or NVDEFINE Setup mode commands.

After a macro is defined, a host program can issue an EXPAND MACRO command, which specifies a macro number. (The operator can use the EXPAND Setup mode command.) The terminal looks up the macro assigned to that macro number and uses that string.

For example, a host program can define macro 301 to be the string, "Type any key to continue." Then, anytime the program needs this string displayed, it could just send an EXPAND MACRO: 301 command. When the terminal receives this command, it looks up macro 301 in its memory and displays the string it finds.

Likewise, the host could send a DEFINE MACRO command to define macro 302 to be a commonly used sequence of commands to the terminal. Thereafter, the host could issue the single command, EXPAND MACRO: 302 whenever it would otherwise send that sequence of commands to the terminal.

Macros are also used another way. Each key or key combination indicates a particular macro number. If a macro is defined for a key's macro number, then when the operator presses that key, the terminal looks up the macro and uses the string of characters it finds instead of the character the key normally produces. A macro that can be expanded by pressing a key is called a *key macro*.

HOST MACROS

Host macros are those macros that can only be expanded by the EXPAND MACRO command. If the macro is a terminal command, the terminal executes that command when the macro is expanded. If the macro is not a command, it is displayed on the screen, just like any message from the host.

Macros are efficient in many operations. For example, instead of sending a long string of characters, the host can send the much shorter EXPAND MACRO command. This saves data transmission time and ensures that the terminal receives the same string each time a particular macro is expanded.

KEY MACROS

Each key or key combination is matched to a macro number (see the detailed description of the DEFINE command). For keys that produce ASCII characters, the associated macro number is the ASCII decimal equivalent of that character. For example, the ASCII decimal equivalent of uppercase P is 80. If macro number number 80 has been defined, pressing uppercase P produces that macro. (If macro number 80 has not been defined, pressing uppercase P just produces the uppercase P character.)

Besides being expanded by a keystroke, key macros can also be expanded by EXPAND MACRO, just like host macros.

Disabling Key Macros

Key macros can be enabled or disabled with the ENABLE KEY EXPANSION command (or the Setup version KEYEXPAND). When key macros are disabled, only their expansion by pressing keyboard keys is disabled; it is still possible to expand them with the EXPAND MACRO command.

Key macros are automatically disabled whenever a SELECT CODE: 2 command or Setup mode CODE EDIT command puts the terminal in Edit mode. (The operator can reenable them with the KEYEXPAND YES Setup mode command.)

LEARN and NVLEARN Commands

You can easily define key macros from the keyboard with the LEARN or NVLEARN Setup mode commands. These commands let you define a key macro by pressing the keys that make up the sequence instead of entering key codes. See the Operators Manual for details.

Keeping a Key Macro Local

Normally, when you press a key, the macro defined for that key is sent to the host computer — just as if you had typed that sequence of characters on the keyboard. However, you might want a key macro only to be displayed on the screen, or you might want a key macro to be interpreted as a command to the terminal. In these cases, you do not want the macro sent to the host, but want it executed locally. To create such a macro, follow these steps:

1. Send the SET KEY EXECUTE CHARACTER command to the terminal. This command defines one character to be the *key execute character*. When this character is in a macro, it means that following characters should be used locally until the key-execute character again appears in the macro.

2. Send the DEFINE MACRO command to define the macro. Put a key-execute character at the beginning and end of the string.

   When the defined key is pressed, the characters enclosed by the key-execute characters are used locally. If these characters are a terminal command, they are executed. If they are not a terminal command, they are displayed as if the operator typed them. The characters are not sent to the host.

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. Macros expanded by the EXPAND MACRO command sent by the host computer are always treated just as if they were sent by the host. (Macros expanded by an EXPAND MACRO command typed by the operator in Local mode, or by a Setup mode KEYEXPAND command, are always treated as if they were typed by the operator.)

CAUTION

When defining a macro to be used locally, be sure to include the second key-execute character. Otherwise, key macros are used locally until the terminal encounters another key-execute character.

VOLATILE AND NONVOLATILE MACROS

The terminal can store a macro in two ways. A *volatile macro* is stored in volatile memory — the macro is not retained when the terminal is turned off or reset. A *nonvolatile macro* is stored in nonvolatile memory, and is remembered even when the terminal is turned off or reset. When the terminal is turned on or reset, all nonvolatile macros are retained.

A macro can be stored in both volatile 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 command (in Setup, DEFINE) defines only the volatile version of a macro. The DEFINE NONVOLATILE MACRO command (in Setup, NVDEFINE) defines 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 (NVSAVE in Setup) before turning off or resetting the terminal.

DEFINING GRAPHTEXT CHARACTERS

You can define your own *stroke precision* graphtext characters. For *string precision* graphtext and alphatext, however, you must use the predefined character character fonts.

GENERAL PROCEDURE

To define your own stroke precision graphtext characters, proceed as follows:

1. Decide on a font number for your stroke precision font. Then delete all characters in that font, and set up a font grid for the characters you will define. For example, if you choose Font 2, the commands might be:

   ```
   DELETE GRAPHTEXT CHARACTER: 2, -1
   SET GRAPHTEXT FONT GRID: 2, 60, 100
   ```

   This deletes all user-defined characters in stroke precision Font 2, and sets up a font grid for defining characters in that font. When defining a character in this example, the model character cell (font grid) extends from X = 0 to X = 60, and from Y = 0 to Y = 100. (These TSU coordinates are only used for defining the stroke precision characters. When stroke precision graphtext is displayed using this font, the model character cell will be scaled to the size set by the SET GRAPHTEXT SIZE command.)

2. Set a pivot point for subsequent graphtext characters. This point in the font grid is used as the lower left corner of the graphtext character. By setting a pivot point higher than Y = 0, you can allow for descenders in lowercase letters.

   ```
   SET PIVOT POINT: (0,20)
   ```

3. Define the individual graphtext characters:

   a. Send BEGIN GRAPHTEXT CHARACTER to start a character definition. For instance,

      ```
      BEGIN GRAPHTEXT CHARACTER: 2, 65
      ```

      starts the definition of character number 65 in stroke precision font number 2.

   b. Draw lines in the model character cell to define the character. Use either the Vector mode or the MOVE-and-DRAW technique for drawing lines.

   c. End the character definition with an END GRAPHTEXT CHARACTER command.

   d. Repeat steps a through c for other stroke precision graphtext characters.

For detailed information, see the descriptions of the SET GRAPHTEXT FONT GRID, BEGIN GRAPHTEXT CHARACTER, END GRAPHTEXT CHARACTER, and DELETE GRAPHTEXT CHARACTER commands.

PIXEL OPERATIONS

INTRODUCTION

Pixel operations give you a much faster way way to place images on the screen or modify images currently there. You can use pixel commands to quickly display and transmit very complex images — much more quickly than with line or panel graphics commands. Unlike line graphics that make up images with a series of MOVEs and DRAWs, pixel operations compose images with groups of pixels.

The disadvantage of pixel operations is that the images are stored only in raster memory; pixel images cannot be stored as segments in the terminal's general memory. Consequently, once you erase the screen, the pixel image is gone; you must retransmit it to display it again. (Segments, on the other hand, can be made invisible and then visible again, with the SET SEGMENT VISIBILITY command.)

A *pixel* is the smallest screen element that a terminal can address. If you display a single pixel on the screen, you will see a tiny dot of color. If all the pixels in a given area are the same color, then the area is that color. But if the pixels in an area alternate between two or more different colors, the area appears to be a color different from any of the individual pixels.

Each pixel on the screen corresponds to a memory location in the terminal's raster memory. (The raster memory was described earlier in this section, in connection with bit planes and surfaces.) As the terminal scans across the raster memory, it reads the number stored in each location. That number represents a color index in the color map. The terminal reads color indices from raster memory, looks in the color map for the entries corresponding to those indices, and displays the pixels using the color mixtures defined by those entries. Thus, the screen display is a visual copy of the contents of raster memory; and by changing the contents of raster memory, you change the image on the screen. One way to do this is to use *pixel operations.*

WAYS OF USING PIXEL OPERATIONS

ALU Mode

All pixel writing operations use an *ALU mode* parameter, set by the BEGIN PIXEL OPERATIONS command. The ALU mode determines how the new color index information in a pixel writing command will affect the pixel information currently stored in raster memory. Depending on the ALU mode, a pixel writing operation may erase the screen, replace one image with another, or combine images. See the BEGIN PIXEL OPERATIONS command's detailed description.

Off-Screen Memory

One feature of pixel operations is the off-screen memory. This is an area of raster memory that is not visible on the screen but can be written to and copied from. You can use off-screen memory as a scratch pad, or as a storage area where you can store images for later retrieval. For example, you could store a special graphics character font in off-screen memory and retrieve each character as it was needed with a PIXEL COPY command.

In the 4107 and 4109 terminals, the on-screen pixel memory extends, in pixel coordinates, from X = 0 to X = 639, and from Y = 0 to Y = 479. The off-screen pixel memory extends from X = 0 to X = 639, and from Y = 480 to Y = 511.

Differences Between Terminals

You should be aware that the results of pixel commands differ significantly between different Tektronix terminals, primarily because of the different dimensions of raster memory. If you write a program for this terminal that uses pixel commands, you may have to rewrite the program in order to make it work with another Tektronix terminal. This is especially true if your program uses the off-screen memory.

WRITING INTO THE PIXEL VIEWPORT

You can access the individual pixels of the terminal's raster memory using the pixel commands. Three of the commands prepare the terminal for writing pixels: BEGIN PIXEL OPERATIONS, SET PIXEL VIEWPORT, and SET PIXEL BEAM POSITION. The pixel-writing commands (RASTER WRITE, RUNLENGTH WRITE, PIXEL COPY, and RECTANGLE FILL) provide varied methods of setting color indices for pixels.

The following examples briefly explain each of these commands, except for PIXEL COPY. (The PIXEL COPY command copies pixels from one rectangular region in raster memory space to another rectangular region.) Refer to the command descriptions in the section *4100-Style Parameter Types, Commands, and Reports* for additional information about these commands.

Figure 3-14 shows commands that define a pixel viewport and write some color indices into that pixel viewport. (The commands are shown both in Setup syntax and in Host syntax.) Let's consider these commands, one by one.

PXBEGIN. In this example, no parameter values are given. Therefore, the PXBEGIN (BEGIN PIXEL OPERATIONS) command assumes default parameters 1, 11, and 6. These parameters specify Graphics Surface 1 as the pixel-writing surface, set the ALU mode to 11 (Replace mode), and specify that 6 bits per pixel be used to transmit each color index.

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 640-by-480 raster memory space. (If you include the off-screen memory, the coordinates would be in "640-by-512" raster 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 Color Index 0.

Setup Command	***Host Syntax***
PXBEGIN	$^{E}_{C}$RU1;6
PXVIEWPORT 100 100 109 109	$^{E}_{C}$RS$^{S}_{P}$e{$^{S}_{P}$[$^{S}_{P}$\`y$^{S}_{P}$Y
PXRECTANGLE 100 100 109 109 0	$^{E}_{C}$RS$^{S}_{P}$e{$^{S}_{P}$[$^{S}_{P}$\`y$^{S}_{P}$Y0
PXPOSITION 3 4	$^{E}_{C}$RH$^{S}_{P}$ca$^{S}_{P}$@
PXRASTERWRITE 4 "4002"	$^{E}_{C}$RP444002
PXPOSITION 3 3	$^{E}_{C}$RH$^{S}_{P}$O\`$^{S}_{P}$@
PXRASTERWRITE 4 "4217"	$^{E}_{C}$RP444217

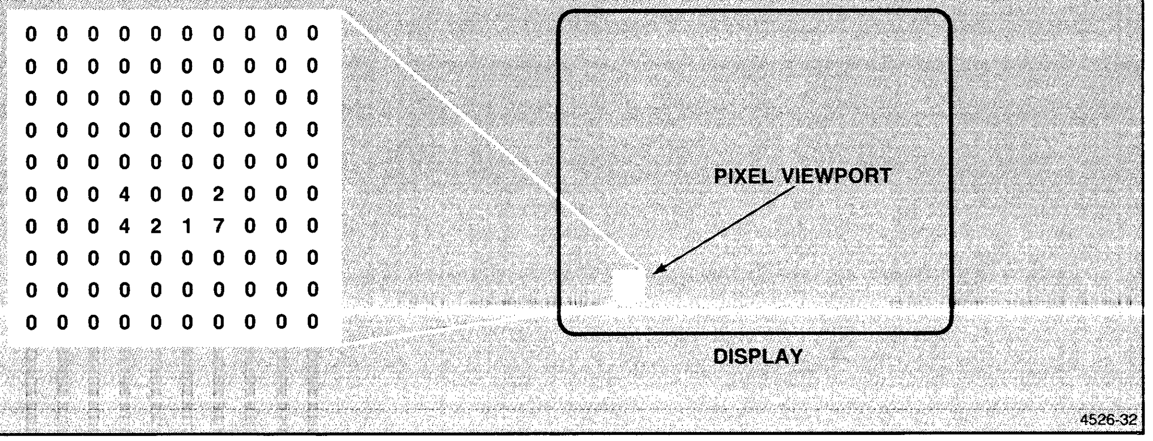


Figure 3-14. Writing Into the Pixel Viewport Using RASTER WRITE.

PXPOSITION 3 4. The PXPOSIITON (SET PIXEL BEAM POSITION) command moves the current pixel position (the point where the next pixel will be written) to the 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 color 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 moves the current pixel position to the location (3,3) in pixel viewport coordinates.

PXRASTEWRWRITE 4 "4217". This writes the color 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.

You can use PXRUNLENGTHWRITE (RUNLENGTH WRITE) commands as well as PXRASTERWRITE (RASTER WRITE) commands to write in the pixel viewport. PXRUNLENGTHWRITE (using an array of specially encoded integers called *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 3-15 explain how this is done.

As in Figure 3-14, Figure 3-15 uses PXBEGIN (this time with bits-per-pixel set to 3) and PXVIEWPORT commands to define a pixel viewport that is ten pixels wide and ten pixels high. As before, a PXRECTANGLE command clears all pixels in the pixel viewport to Color 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.

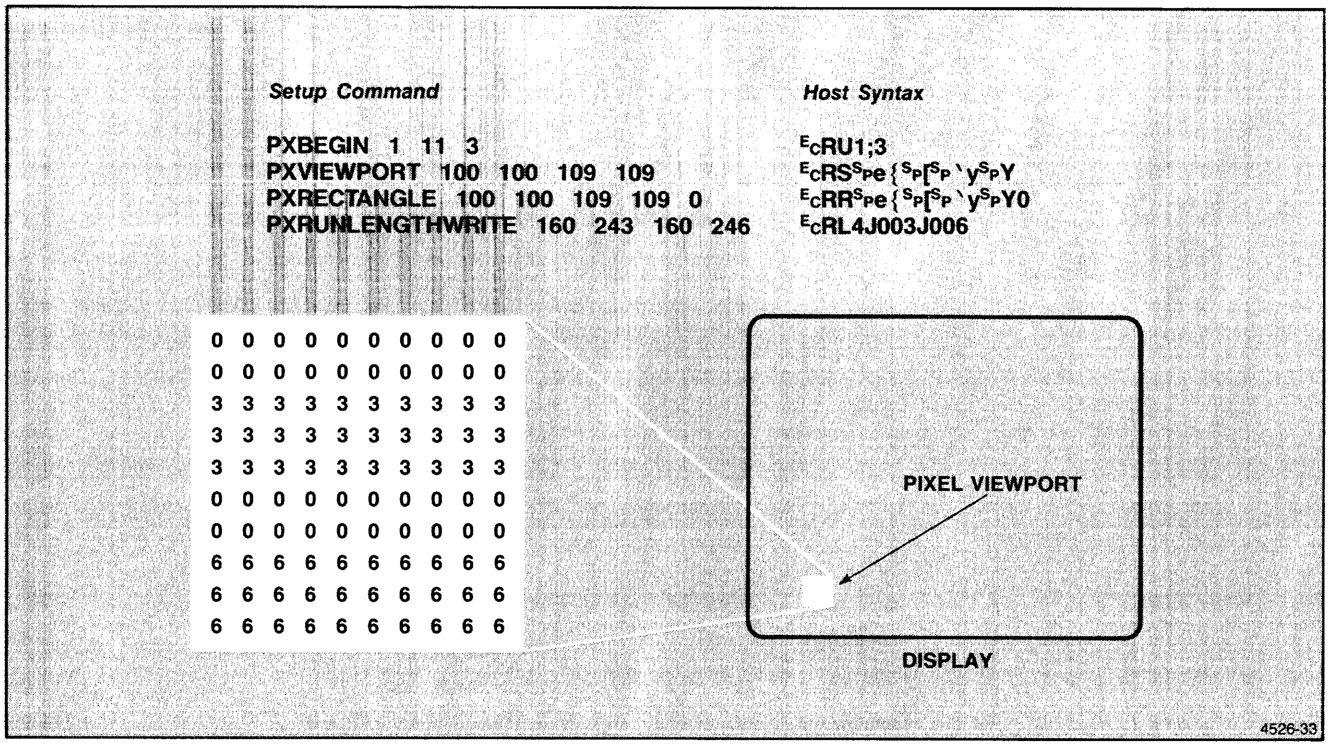
In this example (Figure 3-15), the RUNLENGTH WRITE command has four runcodes in its integer-array parameter. Each runcode is a single number into which two other numbers are packed. The runcodes are packed using the form:

Runcode = number-of-pixels x 2^n + color-index

where n = number-of-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 6; then the value of n is 3.

The first code, 160, calls for 20 pixels of Color Index 0. ($20 \times 2^3 + 0 = 160$.) The second code, 243, calls for 30 pixels of Color Index 3 ($30 \times 2^3 + 3 = 243$.) The third code, 160, calls for 20 pixels of Color Index 0; it is the same as the first code. The fourth code, 246, calls for 30 pixels of Color Index 6 ($30 \times 2^3 + 6 = 246$.)



Setup Command	*Host Syntax*
PXBEGIN 1 11 3	$^{E}{}_{C}$RU1;3
PXVIEWPORT 100 100 109 109	$^{E}{}_{C}$RS$^{S}{}_{P}$e{$^{S}{}_{P}$[$^{S}{}_{P}$`y$^{S}{}_{P}$Y
PXRECTANGLE 100 100 109 109 0	$^{E}{}_{C}$RR$^{S}{}_{P}$e{$^{S}{}_{P}$[$^{S}{}_{P}$`y$^{S}{}_{P}$Y0
PXRUNLENGTHWRITE 160 243 160 246	$^{E}{}_{C}$RL4J003J006

```
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3
3 3 3 3 3 3 3 3 3 3
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6
6 6 6 6 6 6 6 6 6 6
```

Figure 3-15. Writing Into the Pixel Viewport Using RUNLENGTH WRITE.

Section 4

SCREEN EDITING

INTRODUCTION

This section covers the text entry and editing functions of the terminal. The topics covered are:

- *Screen Editing Concepts.* Explains how screen editing differs from line editing and how this terminal's cursor works. Also explains how the dialog area works with screen editing programs.
- *Screen Editing Operating Modes.* Discusses the terminal's editing modes, how they relate to one another, and what functions each mode performs.
- *Ansi and VT52 Mode Command Syntax.* Describes the conventions used in this manual to represent syntax and gives a few tips about syntax rules for these modes.
- *Ansi Mode Commands.* Starts with a list of commands by function and goes on to describe each command in detail.
- *VT52 Mode Commands.* Describes each VT52 mode command in detail.

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 from these standards.

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. The detailed command descriptions provided later in this section explain how the terminal responds to each particular command.

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 modes configure the terminal to run most VT52 or VT100 applications programs.

If you write your own program, it must conform to the commands described in this section.

THE CURSOR

The cursor is an underline displayed on the screen; it 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 *cursor position.*

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

The cursor is displayed on the screen as an underline. You can make the cursor blink and you can choose its color; use the SET ALPHA CURSOR INDEX[1] command. To select the display style and color for text, use the SGR (Select Graphic Rendition) command, described later in this section.

THE DIALOG AREA

The *dialog area* is a display area of the terminal that is separate from the graphics area. All text editing and entry occur in the dialog area, and the Ansi and VT52 mode commands work only in the dialog area. Before executing any screen editing programs, make the dialog area visible with the SET DIALOG AREA VISIBILITY[1] command, or by pressing the Dialog key.

[1] **These are Tek mode commands and are described in the section titled *4100-Style Parameter Types, Commands, and Reports.***

The *dialog buffer* can have as many lines as there is memory to hold them, with up to 32 lines visible on the screen at any one time. (The dialog area is part of the dialog buffer.) You can set the number of lines in the dialog buffer with the SET DIALOG AREA BUFFER SIZE[1] command. You can set the number of lines that are visible with the SET DIALOG AREA LINES[1] command. If the dialog buffer has more than 32 lines, the excess lines are not visible on the screen, but can be scrolled into view. (When the terminal is set to display 132 columns, it can only display 30 lines.)

If the cursor is visible, the dialog buffer rolls up or down as you move the cursor, so that the cursor always remains in view. The exceptions to this are the SU (Scroll Up) and SD (Scroll Down) commands (and the equivalent joydisk movements). These commands scroll the dialog buffer up or down and the cursor maintains its position within the dialog buffer; therefore, the cursor move out of view.

If the cursor is not visible, cursor movement commands do not scroll the dialog buffer.

[1] These are Tek mode commands and are described in the section titled *4100-Style Parameter Types, Commands, and Reports.*

Scrolling Regions

The Ansi mode command set allows the terminal to have up to two *fixed regions* and one *scrolling region* on the screen at the same time. This can be useful when, for example, your screen editing program has a status or message line at the top or bottom of the text editing region; the editing region is the scrolling region and the status lines are the fixed regions.

You can set the *edit margins* of the scrolling region with the TEKSTBM (Set Top and Bottom Margins) command. If the edit margins coincide with the terminal screen size, then there are no fixed regions and the scrolling region is the entire dialog buffer. Otherwise, the lines between and including the edit margins are in the scrolling region and the lines outside the scrolling region are in the fixed regions.

The cursor can be moved from the scrolling region into a fixed region only if Origin mode is set to Absolute (done automatically in Edit mode). Use the CUP (Cursor Position) or HVP (Horizontal and Vertical Position) command to move from one region to the other. Figure 4-1 shows how the dialog buffer and edit margins relate to the terminal display.

Origin mode determines how the cursor is addressed in these fixed and scrolling regions. In Origin mode Absolute, you can address any character position on the screen relative to the upper-left character position of the visible portion of the dialog buffer (character address Row 1, Column 1). The scrolling region and the dialog buffer size are limited to the size of the terminal screen (32 lines maximum). In Origin mode Relative, you can address any character position in the scrolling region relative to the upper-left character position of the scrolling region. If edit margins have not been set, then the scrolling region is the same size as the dialog buffer.

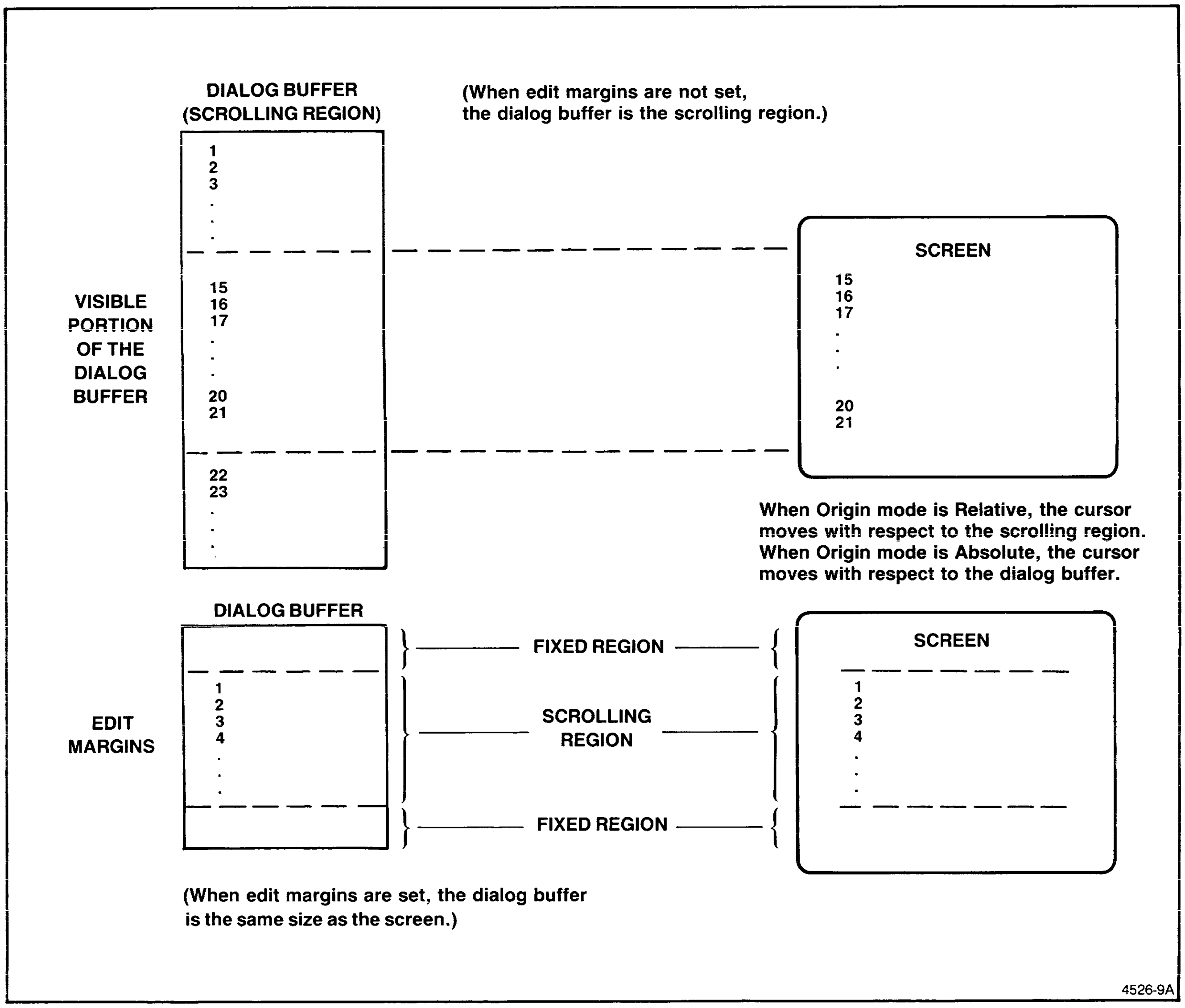

DIALOG BUFFER
(SCROLLING REGION)
(When edit margins are not set,
the dialog buffer is the scrolling region.)
1
2
3
15
16
17
20
21
22
23
VISIBLE PORTION OF THE DIALOG BUFFER
SCREEN
15
16
17
20
21
When Origin mode is Relative, the cursor moves with respect to the scrolling region.
When Origin mode is Absolute, the cursor moves with respect to the dialog buffer.
DIALOG BUFFER
FIXED REGION
SCROLLING REGION
FIXED REGION
SCREEN
1
2
3
4
EDIT MARGINS
(When edit margins are set, the dialog buffer is the same size as the screen.)
4526-9A

Figure 4-1. Fixed and Scrolling Regions in the Dialog Area.

SCREEN EDITING OPERATING MODES

The terminal has two screen editing modes, Ansi mode and VT52 mode. You can configure the terminal to accept VT100 applications by operating in *Edit mode,* a submode of Ansi mode. (More on Edit mode later.) Figure 4-2 shows how these modes relate to one another.

To use a screen editing program:

- The terminal's dialog area must be visible; use the SET DIALOG AREA VISIBILITY command from Setup mode or Tek mode.
- The terminal must be in one of the screen editing modes: use the SELECT CODE command from any mode.

ANSI MODE

In Ansi mode, you can use the RM (Reset Modes) and SM (Set Modes) commands to change the operating characteristics of the terminal. For example, you can:

- Select an 80-column width or 132-column width or 132-column width
- Determine whether characters in the rightmost column automatically wrap to the beginning of the next line
- Determine whether characters are inserted into an existing line or write over existing characters
- Reverse the colors on the display
- Choose whether keyboard keys repeat when held down

Although you set these characteristics while the terminal is in Ansi mode, they stay set when you enter any other mode.

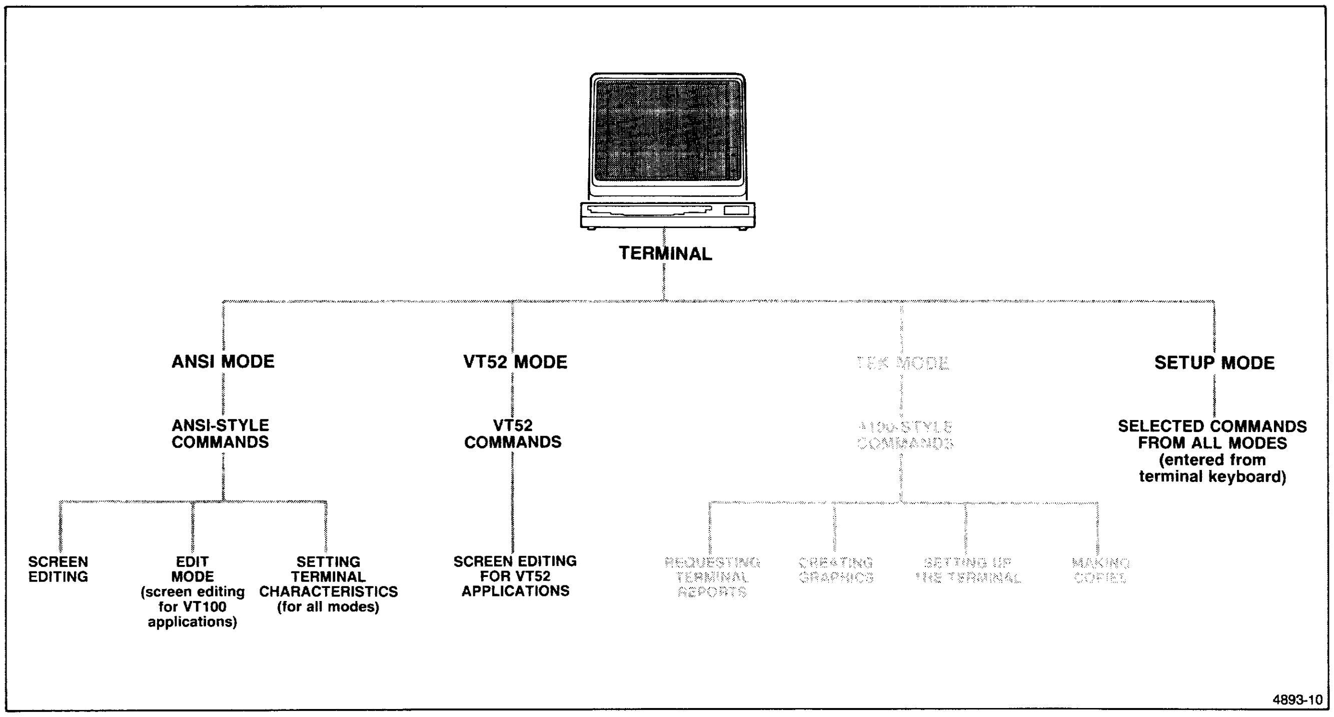

TERMINAL
ANSI MODE
ANSI-STYLE COMMANDS
SCREEN EDITING
EDIT MODE (screen editing for VT100 applications)
SETTING TERMINAL CHARACTERISTICS (for all modes)
VT52 MODE
VT52 COMMANDS
SCREEN EDITING FOR VT52 APPLICATIONS
TEK MODE
4100-STYLE COMMANDS
REQUESTING TERMINAL REPORTS
CREATING GRAPHICS
SETTING UP THE TERMINAL
MAKING COPIES
SETUP MODE
SELECTED COMMANDS FROM ALL MODES (entered from terminal keyboard)
4893-10

Figure 4-2. Screen Editing Modes.

Edit Mode

Edit mode is a submode of Ansi mode, and simply configures the terminal automatically 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 Ansi mode 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 RM and SM command modes:
  - Sets Origin mode to Absolute
  - Sets Insert/Replace mode to Replace (overwrites rather than inserts characters)

Finally, Edit mode temporarily disables all programmed keys. However, you can re-enable their programmed meanings while in Edit mode by using the ENABLE KEY EXPANSION[1] command.

Ansi Mode Commands

Ansi mode commands allow you to:

- Position the cursor
- Set edit margins
- Delete characters and lines
- Erase characters and lines
- Insert characters and lines
- Set the attributes of displayed characters
- Control terminal operating characteristics

Take a look at the list of Ansi mode commands by function immediately preceding their detailed descriptions.

[1] **These are Tek mode commands and are described in the section titled** ***4100-Style Parameter Types, Commands, and Reports.***

VT52 MODE

VT52 mode configures the terminal to run most VT52 applications programs. You can use only the VT52 mode commands listed in this section.

VT52 mode has two submodes: Graphics mode and Alternate Keypad mode. 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.

Entering *Alternate Keypad mode* gives the numeric keypad and Keys F5 through F8 meanings different than their factory default or programmed meanings; exiting this mode restores the factory default or programmed meanings.

If you switch back and forth between Ansi (or Edit) mode and VT52 mode, the numeric keypad retains its mode selection even though the mode name changes. That is, if the keypad is set to send numeric codes while in Ansi mode, then it sends factory default meanings while in VT52 mode; if the keypad is set to send applications codes while in Ansi mode, then it sends the alternate keypad meanings in VT52 mode. See the VT52 command description for ENTER ALTERNATE KEYPAD MODE.

VT52 Mode Commands

VT52 mode commands allow you to:

- Position the cursor
- Erase portions of text
- Control functions and modes

Cursor movement commands move the cursor up, down, left, and right with respect to the dialog buffer.

Erasure commands erase portions of lines or portions of the screen from the current position to the end of the line or bottom of the screen.

Some commands control functions (for example, IDENTIFY). Other commands control modes (like, ENTER GRAPHICS MODE).

ANSI AND VT52 MODE COMMAND SYNTAX

Remember these conventions when using Ansi or VT52 mode commands:

- Command names are always shown in all uppercase characters.
- Characters shown in bold type are those that you must enter exactly as shown.
- Multiword items are joined by hyphens. For example, *parameter-name* and *next-parameter-name* (used in Figure 4-3) represent single items described by two or more words.
- In Host syntax, when a command has more than one parameter, separate them with semicolons.
- In Setup syntax, enter parameters on the same line and separate them with a space or a comma.
- Most commands take integer values for their parameters. The widest valid range is 0 — 32767; however, if you specify a value higher than is reasonable for a particular parameter, the parameter defaults to the highest value that it can accept.
- Default parameter values, if any, are shown at the end of each parameter description.

Figure 4-3 illustrates this section's command description format.

The appendices contain a group of command cross-reference lists. By scanning this appendix, you can, for example, see all the parameter defaults for all commands at a glance. You can locate the descriptive name of a command for which you know the Setup name. You can quickly find a command's opcode, or what its parameters are, or whether its parameters are saved in nonvolatile memory.

COMMAND NAME

The purpose of the command.

Host Syntax

^{E}c[parameter-name; next-parameter-name **Z**

Setup Syntax

SETUPNAME parameter-name, next-parameter-name

parameter-name: Each parameter is explained in a separate paragraph.
Defaults: Omitted = Value if none specified

next-parameter-name: The next parameter explained.
Defaults: Omitted = Value

4893-11

Figure 4-3. Command Description Format for Ansi and VT52 Mode Commands.

ANSI MODE COMMANDS GROUPED BY FUNCTION

Moving the Cursor

CUB — Cursor Backward
CUD — Cursor Down
CUF — Cursor Forward
CUP — Cursor Position
CUU — Cursor Up
HVP — Horizontal and Vertical Position
IND — Index
NEL — Next Line
RI — Reverse Index
TEKRC — Restore Cursor
TEKSC — Save Cursor

Deleting Characters and Lines

DCH — Delete Character
DL — Delete Line

Erasing Characters and Lines

ECH — Erase Character
ED — Erase in Display
EL — Erase in Line

Inserting Characters and Lines

ICH — Insert Character
IL — Insert Line

Choosing Display Styles

SGR — Select Graphic Rendition
TEKDHL — Double Height Line
TEKDWL — Double Width Line
TEKSWL — Single Width Line

Scrolling the Dialog Area

SD — Scroll Down
SU — Scroll Up
SL — Scroll Left
SR — Scroll Right
TEKSTMB — Set Top and Bottom Margins

Working With Tabular Material

CBT — Cursor Backward Tab
CHT — Cursor Horizontal Tab
HTS — Horizontal Tab Set
TBC — Tab Clear
$^{H}{}_{T}$ — Horizontal Tab
$^{V}{}_{T}$ — Vertical Tab

Setting Terminal Characteristics

DMI — Disable Manual Input
EMI — Enable Manual Input
RIS — Reset to Initial State
RM — Reset Modes
SM — Set Modes
TEKKPAM — Keypad Application Mode
TEKKPNM — Keypad Numeric Mode

Changing Modes

SELECT CODE

Control Characters

$^{B}{}_{L}$ Bell
$^{B}{}_{S}$ Backspace
$^{C}{}_{N}$ Cancel
$^{C}{}_{R}$ Carriage Return
$^{F}{}_{F}$ Formfeed
$^{L}{}_{F}$ Linefeed
$^{S}{}_{I}$ Shift In
$^{S}{}_{O}$ Shift Out
$^{S}{}_{P}$ Space
_ Underscore

Reporting to the Host

CPR — Cursor Position Report
DSR — Device Status Report
DA — Device Attributes
REPORT SYNTAX MODE
TEKID — Identify Terminal

ANSI MODE COMMAND DESCRIPTIONS

$^{\mathrm{B}}\mathrm{L}$ (Bell)

Sounds the terminal's bell.

$^{\mathrm{B}}\mathrm{S}$ (Backspace)

Moves the cursor left one position.

The $^{\mathrm{B}}\mathrm{s}$ character moves the cursor one character position to the left. If the cursor is already at Column 1, then $^{\mathrm{B}}\mathrm{s}$ has no effect.

CBT (Cursor Backward Tab)

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

Host Syntax

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

number-of-preceding-tab-stops: Enter 1 to move 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. If there are less than *n* preceding tab stops, the cursor moves to Column 1 of the current line.
Default: Omitted or 0 = 1

CHT (Cursor Horizontal Tab)

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

Host Syntax

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

number-of-following-tab-stops: Enter 1 to move the cursor to the next tab stop. A value greater than 1 (*n*) moves the cursor to the *n*th next tab stop on the current line. If there are less than *n* following tab stops, the cursor moves to the end of the current line.
Default: Omitted or 0 = 1

$^{\mathrm{C}}\mathrm{N}$ (Cancel)

Cancels an Ansi mode command in progress.

CPR (Cursor Position Report)

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

Host Syntax

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

The CPR (Cursor Position Report) message is sent from the terminal to the host in response to a DSR (Device Status Report) command with 6 for a parameter.

If Origin mode is Relative (TEKOM set), the coordinates reported are row, column coordinates in the scrolling region. Row 1, Column 1 is the upper-left corner of the scrolling region.

If Origin mode is Absolute (TEKOM reset), the coordinates reported are row, column coordinates of the dialog buffer. Row 1, Column 1 is the upper-left corner of the dialog buffer.

The terminal does not enter Bypass mode for the CPR.

$^{\mathrm{C}}\mathrm{R}$ (Carriage Return)

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

If Carriage Return/Linefeed ($^{\mathrm{C}}\mathrm{R}^{\mathrm{L}}\mathrm{F}$) mode is set, a linefeed action is also performed.

CUB (Cursor Backward)

Moves the cursor left the specified number of columns.

Host Syntax

^{E}c[number-of-columns **D**

number-of-columns: The number of columns to move the cursor toward the left side of the screen. The cursor does not move beyond Column 1.
Default: Omitted or 0 = 1

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.

CUD (Cursor Down)

Moves the cursor down the specified number of lines.

Host Syntax

^{E}c[number-of-lines **B**

number of lines: The number of lines to move the cursor down.
Default: Omitted or 0 = 1

If Origin mode is Absolute (TEKOM reset), the cursor moves with respect to the dialog buffer.

If Origin mode is Relative (TEKOM set), the cursor moves with respect to the scrolling region.

CUF (Cursor Forward)

Moves the cursor the specified number of columns to the right.

Host Syntax

^{E}c[number-of-columns **C**

number-of-columns: The number of columns to move the cursor toward the right side of the screen. The cursor does not move beyond the rightmost margin.
Default: Omitted or 0 = 1

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.

CUP (Cursor Position)

Moves the cursor to a specified row and column.

Host Syntax

^{E}c[row-number; column-number **H**

row-number: The destination row for the cursor.
Default: Omitted or 0 = 1

column-number: The destination column for the cursor.
Default: Omitted or 0 = 1

If Origin mode is Relative (TEKOM set), the cursor moves with respect to the scrolling region. Row 1, Column 1 is the upper-left corner of the scrolling region.

If Origin mode is Absolute (TEKOM reset), the cursor moves with respect to the dialog buffer. Row 1, Column 1 is the upper-left corner of the dialog buffer.

CUU (Cursor Up)

Moves the cursor upward the specified number of lines.

Host Syntax

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

number-of-lines: The number of lines to move the cursor toward the top of the screen.
Default: Omitted or 0 = 1

If Origin mode is Absolute (TEKOM reset), the cursor moves with respect to the dialog buffer.

If Origin mode is Relative (TEKOM set), the cursor moves with respect to the scrolling region.

DA (Device Attributes)

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

Host Syntax

```
Ec[ device-status-request c
```

device-status-request: This parameter is always 0.
Default: Omitted = 0

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 echos this report back to the terminal, the terminal ignores the echo.

DCH (Delete Character)

Deletes the character at the cursor and characters following the cursor depending on the value of the parameter.

Host Syntax

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

number-of-characters: The number of characters to delete.
Default: 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.

DL (Delete Line)

Deletes the specified number of lines starting with the current line.

Host Syntax

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

number-of-lines: The number of lines to delete.
Default: Omitted or 0 = 1

All lines following the deleted lines (including those in the invisible portion of the scroll) are shifted in a block toward the line containing the cursor. The cursor does not change position. If fixed and scrolling regions have been defined, this command only affects lines in the region that the cursor is currently in. For example, if the cursor is in the top fixed region, only the lines in the top fixed region are affected.

DMI (Disable Manual Input)

Disables the keyboard.

Host Syntax

$^{E}{}_{C}$`

This command is the equivalent of the Ansi mode SM command with the KAM parameter or of the 4100-style command LOCK KEYBOARD with a parameter of 1.

DSR (Device Status Report)

Generates a CPR (Cursor Position Report) or a DSR (Device Status Report).

Host Syntax

$^{E}{}_{C}$[status **n**

status: Determines whether the DSR is a command from the host or a report from the terminal (see Table 4-1 for parameter values).

When the host sends a DSR command with a parameter of 5, the terminal responds with a DSR message telling whether or not there is a malfunction. When the host sends a DSR command with a parameter of 6, the terminal responds with a CPR (see CPR command for the meaning of the response). If the terminal receives a DSR with a parameter value of 0 or 3 (which could be the echo of a report it has sent to the host), the terminal ignores that DSR.

Table 4-1

DEVICE STATUS REPORT PARAMETERS

Parameter	Meaning
0	Report from terminal: ready — no malfunctions detected
3	Report from terminal: malfunction — try again
5	Command from host: report status using a DSR
6	Command from host: report cursor position using a CPR (see CPR command)

ECH (Erase Character)

Erases the specified number of characters starting at the cursor position.

Host Syntax

$^{E}{}_{C}$[number-of-characters **X**

number-of-characters: The number of characters to erase.
Default: 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.

ED (Erase in Display)

Erases part or all of the current display, according to the parameter.

Host Syntax

$^{E}{}_{C}$[erase-extent **J**

erase-extent: Enter 0 to erase text from the cursor position to the end of the dialog buffer. Enter 1 to erase text from the beginning of the dialog buffer to the cursor position. Enter 2 to erase the entire dialog buffer.
Default: Omitted = 0

The cursor does not change position.

EL (Erase in Line)

Erases part or all of the current line, according to the parameter.

Host Syntax

```
EC[ erase-extent K
```

erase-extent: Enter 0 to erase text from the cursor position to the end of the line. Enter 1 to erase text from the beginning of the line to the cursor position. Enter 2 to erase the entire line.
Default: Omitted = 0

EMI (Enable Manual Input)

Enables the keyboard.

Host Syntax

```
ECb
```

This command is the equivalent of the ANSI-style RM command with the KAM parameter or of the 4100-style LOCK KEYBOARD command with a parameter of 0.

FF (Formfeed)

The terminal inserts FF into the dialog area and advances the cursor position. This character indicates the start of a new page to a hardcopy unit (see command description for Tek mode command SET DIALOG HARDCOPY ATTRIBUTES).

HT (Horizontal Tab)

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

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

The tab stops that are in effect when the terminal is turned on are at columns: 1, 9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 97, 105, 113, 121, 129. These power-up tab stops can be changed and saved with the SAVE NONVOLATILE PARAMETERS command.

HTS (Horizontal Tab Set)

Sets a tab stop at the current cursor location or sets the tab stops for the entire screen.

Host Syntax

```
ECH
```

Setup Syntax

```
TABS list-of-tab-stops
```

list-of-tab-stops: A list of numbers representing the column numbers where you want tab stops set.
Default: Factory default = 1, 9, 17, 25, 33, 41, 49, 57, 65, 73, 81, 89, 97, 105, 113, 121, 129.

The tab stops that are in effect when the terminal is turned on are those stops that have been saved in nonvolatile memory.

HVP (Horizontal and Vertical Position)

Moves the cursor to a specified row and column of the screen.

Host Syntax

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

row-number: The destination horizontal row number for the cursor.
Default: Omitted or 0 = 1

column-number: The destination vertical column number for the cursor.
Default: Omitted or 0 = 1

If Origin mode is Relative (TEKOM set), the cursor moves with respect to 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 moves with respect to the dialog buffer. "Row 1, Column 1" is the upper-left corner of the dialog buffer. The cursor does not move beyond the top and bottom of the dialog buffer.

ICH (Insert Character)

Inserts the specified number of Space characters at the cursor position.

Host Syntax

```
EC[ number-of-characters @
```

number-of-characters: The number of Space characters to insert.
Default: 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.

IL (Insert Line)

Inserts the specified number of blank lines in place of the current line.

Host Syntax

```
EC[ number-of-lines L
```

number-of-lines: The number of lines to insert.
Default: Omitted or 0 = 1

The current line and all following lines are shifted down. 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 that the cursor is currently in (if the cursor is in the scrolling (nonfixed) region, only the lines in the scrolling region are affected).

IND (Index)

Moves the cursor down one line without affecting its character 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. The cursor can index into the scrolling region from the top fixed region, but cannot index into the bottom fixed region. An index on the last line of the bottom fixed region has no effect.

LF (Linefeed)

Moves the cursor position down one line.

If LNM (Linefeed/Newline mode) is reset, 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, then LF has the same effect as CR IND: it advances the cursor position to the first character position on the following line.

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 CRIND (a Carriage Return character followed by an IND (Index) command); the cursor moves to the first character position on the next line.

REPORT SYNTAX MODE

Sends a Terminal Settings Report to the host.

Host Syntax

```
EC#!0
```

The Terminal Settings Report contains the terminal's syntax mode status.

This command has the same effect as a REPORT TERMINAL SETTINGS command issued for the SELECT CODE command (as if EC!Q%! was sent from the host). See the REPORT TERMINAL SETTINGS command in Section 5 for additional information.

This command is recognized in all 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 condition.

Host Syntax

```
ECc
```

When the terminal receives this command, it:

- Erases the screen
- Positions the dialog area cursor at the Home position
- Sets Insert/Replace mode to Replace
- Clears edit margins
- Turns off the character graphic rendition feature
- Selects default G0 or G1 character set
- Shifts in the G0 character set
- Enables the dialog area
- Makes the dialog area visible

RM (Reset Modes)

Resets one or more terminal modes.

Host Syntax

```
EC[ mode l
```

Each mode is specified by a separate parameter. You can reset more than one mode by stringing parameters together, separated by semicolons. A mode is reset until you set it again with the SM (Set Modes) command.

NOTE

If you are resetting more than one mode, put parameters that contain digits **only** *first, followed by parameters containing special characters (***?** *and* **<***). When a* **?** *is the first character in the parameter list, the terminal interprets all subsequent digit-only parameters as also beginning with a* **?** *character.*

Alternatively, if you're issuing a series of parameters that all start with the **?** *character, you can issue the first parameter as a* **?** *only, followed by just the digit of each parameter you issue.*

The following paragraphs describe each mode's set and reset condition; Table 4-2 (next page) summarizes this same information and gives the parameters that set and reset each terminal mode.

Keyboard Action Mode (KAM). When reset, enables the terminal keyboard. When set, disables the terminal keyboard.

Insert/Replace Mode (IRM). When reset, each entered character overwrites (that is, *replaces*) the character at the cursor position. When set, each entered character is inserted at the cursor position and characters at and to the right of the cursor position move to the right.

Send/Receive Mode (SRM). When reset, the terminal displays characters as they are sent to the host (local echo). When set, the terminal does not echo locally.

Linefeed/Newline Mode (LNM). When reset, LF (Linefeed) moves the cursor down one line without changing its column position. When set, LF moves the cursor to the beginning of the next line.

Overstrike/Replace Mode. When reset, SP (Space) and (Underscore) are treated as normal characters. When set, SP advances the cursor to the next character position and underlines the current character.

Cursor Keys Mode (TEKCKM). When reset, Function Keys F1 — F4 transmit regular ANSI cursor control commands; however, if these keys are programmed and key expansions are enabled, they transmit their programmed values. When set, Function Keys F1 — F4 transmit application program codes. Table 4-3 shows the codes that these keys transmit when reset and set.

Ansi-to-VT52 Mode (TEKANM). When reset, puts the terminal in VT52 mode. Has no effect when set.

Column Mode (TEKCOLM). When reset, specifies 80-column width. When set, specifies 132-column width. When Column mode is set to 132, the maximum number of visible lines in the dialog area is reduced from 32 to 30.

Setting and resetting this mode erases the contents of the dialog area and resets margins, but 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, but horizontal scrolling can bring any of the columns into view. If you make the dialog area visible with the Dialog key, the dialog area will scroll automatically. That is, if the cursor is in Columns 1 — 80, the dialog area will scroll right to bring the leftmost column into view; if the cursor is in Columns 81 — 132, the dialog area will scroll left to bring the rightmost column into view.

Screen Mode (TEKSCNM). When reset, reverses the colors on the display from reversed to normal, and Index 0 in the dialog area changes from transparent to opaque. When set, reverses the colors on the display from normal to reversed, and Index 0 in the dialog area returns to transparent.

Origin Mode (TEKOM). When reset, Origin mode is Absolute; the cursor moves to Row 1, Column 1 of the dialog area and the dialog buffer size is reduced to the screen size. When set, Origin mode is Relative and the cursor moves to Row 1, Column 1 of the scrolling region.

Autowrap Mode (TEKAWM). When reset, disables autowrap so that characters entered in the rightmost column write over existing characters in that column. When set, enables autowrap so that characters entered in the rightmost column wrap around to the next line.

Autorepeat Mode (TEKARM). When reset, disables autorepeat so that keyboard keys do not repeat when held down. When set, enables autorepeat so that keyboard keys repeat when held down.

Table 4-2

RM (RESET MODES) AND SM (SET MODES) COMMAND PARAMETERS

Host Parameter	Mode Name	Result	Setup Version
2	KAM (Keyboard Action mode)	Reset: Enabled Set: Disabled	LOCKKEYBOARD no LOCKKEYBOARD yes
4	IRM (Insert/ Replace mode)	Reset: Replace Set: Insert	INSERTREPLACE replace INSERTREPLACE insert
12	SRM (Send/ Receive mode)	Reset: Enables echo Set: Disables echo	ECHO yes ECHO no
20	LNM (Linefeed/ Newline mode)	Reset: Linefeed Set: Newline	LFCR no LFCR yes
<1	TEKORM (Overstrike/ Replace mode)	Reset: Overstrike Set: Replace	DAMODE overstrike DAMODE replace
?1	TEKCKM (Cursor Keys mode)	Reset: Keys F1 — F4 normal Set: Keys F1 — F4 application	CURSORKEYMODE no CURSORKEYMODE yes
?2	TEKANM (Ansi-to-VT52 mode)	Reset: VT52 mode Set: No effect	CODE VT52
?3	TEKCOLM (Column mode)	Reset: 80 columns Set: 132 columns	COLUMNMODE 80 COLUMNMODE 132
?5	TEKSCNM (Screen mode)	Reset: Normal display Set: Reversed display	SCREENMODE normal SCREENMODE reverse
?6	TEKOM (Origin mode)	Reset: Absolute Set: Relative	ORIGINMODE absolute ORIGINMODE relative
?7	TEKAWM (Autowrap mode)	Reset: Disables autowrap Set: Enables autowrap	AUTOWRAP no AUTOWRAP yes
?8	TEKARM (Autorepeat mode)	Reset: Disables autorepeat Set: Enable autorepeat	AUTOREPEAT no AUTOREPEAT yes

Table 4-3

CURSOR KEY MODE CODES

Function Key	Codes Sent When Set	Codes Sent When Reset
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

${}^{S}{}_{B}$ (Sub)

Cancels an Ansi mode command in progress.

SCS (Select Character Set)

Selects default and special character sets.

Host Syntax (to select G0)

```
Ec( character-set
```

Host Syntax (to select G1)

```
Ec) character-set
```

Setup Syntax (to select G0)

```
SELECTCHARSET G0 character-set
```

Setup Syntax (to select G1)

```
SELECTCHARSET G1 character-set
```

character-set: The parameter from Table 4-4 that designates the desired character set as either the G0 or G1 character set.

The terminal allows you to access two different character sets by using the ${}^{S}{}_{I}$ (selects the current G0 character set) and ${}^{S}{}_{O}$ (selects the current G1 character set) commands. First you must designate which of the eight possible character sets you want to use and then you can easily switch between them.

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

Table 4-4 shows the parameters needed to select a particular character set. The appendices contain tables that list the contents of each character set.

Table 4-4

SCS (SELECT CHARACTER SET) COMMAND PARAMETERS

Character Set	Character Set Designated
A	United Kingdom
B	U.S. (ASCII)
G	Swedish
K	German
f	French (See note)
`	Danish/Norwegian
0	Ruling Set
3	Supplementary

NOTE

*Please use f to designate the French character set. These are the current standard escape sequence parameters. However, for compatability with programs designed for an earlier version of the French character set, the terminal does accept **R** as a synonym for f.*

SD (Scroll Down)

Scrolls lines down.

Host Syntax

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

number-of-lines: The number of lines to scroll toward the bottom of the screen.
Default: Omitted or 0 = 1

The SD command shifts all lines in the dialog buffer 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.

SELECT CODE

Causes the terminal to recognize Ansi, Tek, or VT52 mode command syntax. Also used to select Edit mode.

Host Syntax

```
Ec%! syntax
```

Setup Syntax

```
CODE syntax
```

syntax: The mode in which you want to operate. An integer in Host syntax or a keyword in Setup syntax:

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

Default: Omitted = Tek mode

The syntax of Tek, Ansi, and VT52 mode commands are not compatible. If you are using 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 major modes: Ansi, Setup, Tek, and VT52.

See the beginning of this section for an explanation of Edit mode.

SGR (Select Graphic Rendition)

Invokes the character display style specified by the parameters.

Host Syntax

```
Ec[ graphic-rendition m
```

Setup Syntax

```
TEXTRENDITION graphic-rendition
```

graphic-rendition: The style in which text is displayed in the dialog area.
Default: Omitted = 0

In Host syntax, you can set more than one display attribute by entering more than one parameter separated by semicolons. In Setup mode, parameters must be separated by one or more spaces. If you are entering more than one parameter, any parameters that begin with **?**, <, =, or > should be at the start of the list of parameters. Refer to Table 4-5 for an explanation of these parameters. All characters after you send the SGR command are displayed as specified by the SGR parameters until you execute another SGR command.

In Table 4-5, *foreground index* is the color index of the characters as they are displayed on the screen. *Background index* is the color of the screen on which the characters are displayed. *Erase index* is the color with which the characters are erased when a delete or erase command is issued.

Table 4-5

SGR (SELECT GRAPHIC RENDITION) COMMAND PARAMETERS

Parameter	Description
0	Characters displayed in specified color indices with all other attributes (blink, bold, underscore, reverse) turned off.
1	Bold or increased intensity. Bold text is displayed in Color Index 2. Color Index 2 defaults to red. Character background remains the same as current background index.
4	Underscore text.
5	Slow blink (less than 150 blinks per minute).
7	Reverse video text. Foreground and background color indices are swapped.
24	No underscore. Cancels the effect of SGR with *graphic-rendition* set to 4.
25	No blink. Cancels the effect of SGR with *graphic-rendition* set to 5.
27	Positive video text. Cancels the effect of SGR with *graphic-rendition* set to 7.
30	Black display. Sets current character color index to 0, which defaults to black.
31	Red display. Sets current character color index to 2, which defaults to red.
32	Green display. Sets current character color index to 3, which defaults to green.
33	Yellow display. Sets current character color index to 7, which defaults to yellow.
34	Blue display. Sets current character color index to 4, which defaults to blue.
35	Magenta display. Sets current character color index to 6, which defaults to magenta.
36	Cyan display. Sets current character color index to 5, which defaults to cyan.
37	White display. Sets current character color index to 1, which defaults to white.
39	Default display color, Color Index 1, which defaults to white.

Table 4-5 (cont)

SGR (SELECT GRAPHIC RENDITION) COMMAND PARAMETERS

Parameter	Description
40	Transparent background. Sets current character background color index to 0, which defaults to transparent.
41	Red background. Sets current character background color index to 2, which defaults to red.
42	Green Background. Sets current character background color index to 3, which defaults to green.
43	Yellow background. Sets current character background color index to 7, which defaults to yellow.
44	Blue background. Sets current character background color index to 4, which defaults to blue.
45	Magenta background. Sets current character background color index to 6, which defaults to magenta.
46	Cyan background. Sets current character background color index to 5, which defaults to cyan.
47	White background. Sets current character background color index to 1, which defaults to white.
49	Default background color, Color Index 0, which defaults to transparent.
< *index*	Character color. Parameter specifies the color index. Indices from 0 to 7 select colors in the terminal's color map. Indices greater than 7 default to 7.
= *index*	Background color. Parameter specifies the character background color index. Index 0 means that the graphics area shows through. Background indices from 1 to 7 represent colors in the terminal's color map. Indices greater than 7 default to 7.
> *index*	Erase color. Parameter specifies the dialog area erase color index. Erase Index 0 always means that the graphics area shows through. Indices from 1 to 7 select colors in the terminal's color map. Indices greater than 7 default to 7.

SI (Shift In)

Invokes the current G0 character set.

The terminal allows you to access two different character sets by using the SI (selects the current G0 character set) and SO (selects the current G1 character set) commands.

The SI command invokes the current G0 character set installed in the terminal. 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; each of these character sets is designated by connecting a standard or optional keyboard to the terminal or by using the SCS (Select Character Set) command. Appendix A of this manual lists these character sets.

To select the G1 character set, use the SO (Shift Out) character.

SL (Scroll Left)

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

Host Syntax

```
EC[ number-of-columnsSP@
```

number-of-columns: Specifies the number of columns to scroll left.
Default: Omitted or 0 = 1

The SL command moves the entire contents of the visible 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. 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 SCROLL LEFT or SCROLL RIGHT command (or you can use the joydisk). However, an automatic scroll left may occur when you make the dialog area visible with the DIALOG key; if the cursor is in Columns 81 — 132, the terminal will scroll left to bring the rightmost column into view.

SM (Set Modes)

Sets one or more modes.

Host Syntax

```
EC[ mode h
```

mode: The terminal mode or modes that you want to set.
Default: Omitted = 0 (not a valid parameter)

Each mode to be set is specified by a separate parameter. You can set one or more modes by entering more than one parameter separated by semicolons. A mode remains set until you reset it with an RM (Reset Mode) command. Refer back to the RM command description for an explanation of the *mode* parameters.

SO (Shift Out)

Invokes the G1 character set.

The terminal allows you to access two different character sets by using the SI (selects the current G0 character set) and SO (selects the G1 character set) commands.

The SO command invokes the 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) character.

SP (Space)

Moves the cursor one character position to the right or inserts a SP (Space) character.

If Overstrike mode is enabled (TEKORM is reset), SP moves the cursor one character to the right. The SP character does not replace the character, if any, at the cursor position.

If Replace mode is enabled (TEKORM is set), SP replaces any character at the cursor position.

SR (Scroll Right)

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

Host Syntax

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

number-of-columns: Specifies the number of columns to scroll right.
Default: Omitted or 0 = 1

The SR command moves the entire contents of the visible 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 SCROLL LEFT or SCROLL RIGHT command (or you can use the joydisk). However, an automatic scroll right may occur when you make the dialog area visible with the DIALOG key; if the cursor is in Columns 1 — 80, the terminal will scroll right to bring the rightmost column into view.

SU (Scroll Up)

Scrolls lines up.

Host Syntax

```
EC[ number-of-lines S
```

number-of-lines: The number of lines to scroll toward the top of the screen.
Default: Omitted or 0 = 1

The SU (Scroll Up) command shifts all lines in the dialog buffer 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.

TBC (Tab Clear)

Clears one or more tab stops.

Host Syntax

```
EC[ tab-clear-extent g
```

tab-clear-extent: Clears the tabs as indicated by the parameter values. Refer to Table 4-6 for an explanation of these parameters.
Default: Omitted = 0

Table 4-6

TBC (TAB CLEAR) COMMAND PARAMETERS

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

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

Top Half	**Bottom Half**
^{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. Making a dialog copy 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 Tek mode HARDCOPY command description 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 Tek mode HARDCOPY command description 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}cZ

This command causes the same response as the Ansi mode DEVICE ATTRIBUTES (DA) command 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)

Places the terminal in Keypad Application mode.

Host Syntax

```
Ec=
```

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 4-7 for an explanation of these codes.

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

TEKKPNM (Keypad Numeric Mode)

Places the terminal in Keypad Numeric mode.

Host Syntax

```
Ec>
```

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 4-7. 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 Keypad Numeric mode (keys produce their default meanings).

Table 4-7

NUMERIC KEYPAD PROGRAMMING CODES

Numeric Keypad Key	Characters Sent in Application Mode	Characters Sent in Numeric Mode
0	EcOp	0
1	EcOq	1
2	EcOr	2
3	EcOs	3
4	EcOt	4
5	EcOu	5
6	EcOv	6
7	EcOw	7
8	EcOx	8
9	EcOy	9
-	EcOm	-
,	EcOl	,
.	EcOn	.
ENTER	EcOM	CR
F5	EcOP	EcOP
F6	EcOQ	EcOQ
F7	EcOR	EcOR
F8	EcOS	EcOS

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, then TEKRC (Restore Cursor) returns the cursor to the Home position 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 saves information about the cursor position, graphic rendition, character set, and Origin mode in the terminal's memory. This saved information may be restored using the TEKRC (Restore Cursor) command.

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: The row number of the top margin.
Default: Omitted or 0 = 1

bottom-margin: The row number of the bottom margin.
Default: Omitted or 0 = The number of lines visible in the dialog buffer (DALINES).

The parameter 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.

TEKSWL (Single Width Line)

Causes the line containing the cursor 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)

The VT character moves the cursor down one line without affecting the cursor position on the line.

_ (Underscore)

If Overstrike mode is enabled (TEKORM is reset), the current character is underlined and the cursor advances to the next character.

If Replace mode is enabled (TEKORM is set), the underscore character replaces the current character.

VT52 COMMAND DESCRIPTIONS

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 mode
- Sending an RM command (**$^E{}_C$[?2l**) from the host while in Ansi mode
- Sending a SELECT CODE command ($^E{}_C$**%!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 and REPORT SYNTAX MODE, both of which work in all modes.

CURSOR DOWN

Moves the cursor down one line without moving it horizontally.

Host Syntax

```
EcB
```

The cursor moves with respect to the dialog buffer and 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 mode 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 mode 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 moves with respect to the dialog buffer and 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

```
ᴇcY line; column
```

line: An ASCII character that represents the line position number plus 31. The maximum line range is 96, even if the dialog buffer is larger.

column: An ASCII character that represents the column number plus 31. The maximum column range is 80, even if Column mode is set to 132.

For example, to move the cursor to Line 3, Column 1, give the command $^{E}_{C}$**Y**"$^{S}_{P}$ since the decimal equivalent of " is 34 (3 + 31) and the decimal equivalent of $^{S}_{P}$ is 32 (1 + 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.

ENTER ALTERNATE KEYPAD MODE

Causes the numeric keypad keys and Function Keys F5 — F8 to assume their Alternate Keypad mode meanings (shown in Table 4-8).

Host Syntax

$^{E}_{C}$=

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

This command works like the Ansi mode command TEKKPAM (Keypad Application Mode).

Table 4-8

ALTERNATE KEYPAD PROGRAMMING CODES

Numeric Keypad Key	Characters Sent as Factory Default	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	$^{C}_{R}$	$^{E}_{C}$?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

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

Causes the Rulings character set to be selected 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 mode 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 mode 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 — F8 to assume their factory default meanings, or their programmed meanings if they have been programmed.

Host Syntax

$^{E}{}_{C}$>

Factory default meanings are shown in Table 4-8 (under ENTER ALTERNATE KEYPAD MODE).

This command works like the Ansi mode command TEKKPNM (Keypad Numeric Mode).

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

$^{E}{}_{C}$Z

When the host issues this command, the terminal sends its identifier escape 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
```

This command has the same effect as a REPORT TERMINAL SETTINGS command issued for the SELECT CODE command (as if ^{E}c**IQ%!** were sent from the host). See the REPORT TERMINAL SETTINGS command in Section 5 for additional information.

This command is recognized in all modes: Ansi, Edit, Tek, and VT52.

REVERSE LINEFEED

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

Host Syntax

```
Ec I
```

SELECT CODE

Causes the terminal to recognize Ansi, Tek, or VT52 mode command syntax. Also used to select Edit mode.

Host Syntax

```
Ec%! syntax
```

Setup Syntax

```
CODE syntax
```

syntax: The mode in which you want to operate. An integer in Host syntax or a keyword in Setup syntax:

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

Default: Omitted = Tek mode

The syntax of Tek, Ansi, and VT52 mode commands are not compatible. If you are using 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.

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 major modes: Ansi, Setup, Tek, and VT52.

4100-STYLE COMMANDS BY FUNCTION

CONTROLLING THE DISPLAY

4010 HARDCOPY
CRLF
ENABLE DIALOG AREA
HARDCOPY
LFCR
PAGE
RENEW VIEW
SET ECHO
SET ERROR THRESHOLD
SET FIXUP LEVEL
SET GRAPHICS AREA WRITING MODE
SET SNOOPY MODE
SET TAB STOPS

DIALOG AREA

CLEAR DIALOG SCROLL
ENABLE DIALOG AREA
SET DIALOG AREA BUFFER SIZE
SET DIALOG AREA COLOR MAP
SET DIALOG AREA INDEX
SET DIALOG AREA LINES
SET DIALOG AREA VISIBILITY
SET DIALOG AREA WRITING MODE
SET DIALOG HARDCOPY ATTRIBUTES

GENERAL TERMINAL CONTROL

CANCEL
FACTORY
HELP
IGNORE DELETES
LOCAL
MACRO STATUS
RESET
SET EDIT CHARACTERS
STATUS

GIN (GRAPHIC INPUT)

Enabling and Disabling GIN

DISABLE GIN
ENABLE 4010 GIN
ENABLE GIN
REPORT GIN POINT

Setting GIN Parameters

SET GIN AREA
SET GIN CURSOR
SET GIN CURSOR COLOR
SET GIN DISPLAY START POINT
SET GIN GRIDDING
SET GIN INKING
SET GIN RUBBERBANDING
SET GIN STROKE FILTERING
SET GIN WINDOW
SET PICK APERTURE

Setting Parameters for GIN Reports

SET REPORT EOM FREQUENCY
SET REPORT MAX LINE LENGTH
SET REPORT SIG CHARACTERS

GRAPHICS COLORS

SET BACKGROUND COLOR
SET BACKGROUND INDICES
SET COLOR MODE
SET LINE INDEX
SET SURFACE COLOR MAP
SET TEXT INDEX

GRAPHICS PRIMITIVES AND PRIMITIVE ATTRIBUTES

Alphatext

ENTER ALPHA MODE
SET ALPHA CURSOR INDICES
SET ALPHATEXT FONT
SET TEXT INDEX
SET 4014 ALPHATEXT SIZE

Graphtext

GRAPHIC TEXT
SET CHARACTER PATH
SET GRAPHTEXT FONT
SET GRAPHTEXT FONT GRID
SET GRAPHTEXT PRECISION
SET GRAPHTEXT ROTATION
SET GRAPHTEXT SIZE
SET GRAPHTEXT SLANT
SET TEXT INDEX

Lines

DRAW
ENTER VECTOR MODE
MOVE
SET 4014 LINE STYLE
SET LINE INDEX
SET LINE STYLE

Markers

DRAW MARKER
ENTER MARKER MODE
SET MARKER TYPE

Other Graphics Primitives

INCLUDE COPY OF SEGMENT
SET GRAPHICS AREA WRITING MODE
SET PICK ID

Panels

BEGIN PANEL BOUNDARY
END PANEL
SELECT FILL PATTERN

GRAPHTEXT

Defining Graphtext Characters

BEGIN GRAPHTEXT CHARACTER
DELETE GRAPHTEXT CHARACTER
END GRAPHTEXT CHARACTER
SET GRAPHTEXT FONT GRID

Displaying Graphtext

GRAPHIC TEXT
SET GRAPHTEXT FONT
SET GRAPHTEXT PRECISION
SET GRAPHTEXT ROTATION
SET GRAPHTEXT SIZE
SET GRAPHTEXT SLANT

HARDCOPIES

COPY
HARDCOPY
SELECT HARDCOPY INTERFACE
SET COPY SIZE
SET DIALOG HARDCOPY ATTRIBUTES
4010 HARDCOPY

HOST PORT COMMUNICATIONS

COPY
ENTER BYPASS MODE
IGNORE DELETES
PROMPT MODE
SET BAUD RATES
SET BREAK TIME
SET BYPASS CANCEL CHARACTER
SET EOF STRING
SET EOL STRING
SET EOM CHARACTERS
SET FLAGGING MODE
SET PARITY
SET PROMPT STRING
SET QUEUE SIZE
SET REPORT MAX LINE LENGTH
SET STOP BITS
SET TRANSMIT DELAY
SET TRANSMIT RATE LIMIT

KEYBOARD CONTROL AND MACRO DEFINITIONS

DEFINE MACRO
EXPAND MACRO
LEARN/NVLEARN
LOCK KEYBOARD
LOCK VIEWING KEYS
MACRO STATUS
SET KEY EXECUTE CHARACTER
SET TAB STOPS

MODE SWITCHING

ENTER ALPHA MODE
ENTER BYPASS MODE
ENTER MARKER MODE
ENTER VECTOR MODE
SELECT CODE

MULTIPLE VIEWS

Controlling a Viewports Border

SET BORDER VISIBILITY

Defining and Selecting Views

DELETE VIEW
RENEW VIEW
SELECT VIEW
SET VIEW ATTRIBUTES
SET VIEW DISPLAY CLUSTER
SET VIEWPORT
SET WINDOW

Zoom and Pan Operations

LOCK VIEWING KEYS
SET WINDOW

NON-GIN REPORTS

Requesting Non-GIN Reports

REPORT 4010 STATUS
REPORT DEVICE STATUS
REPORT ERRORS
REPORT PORT STATUS
REPORT SEGMENT STATUS
REPORT SYNTAX MODE
REPORT TERMINAL SETTINGS

Setting Parameters for Non-GIN Reports

SET REPORT EOM FREQUENCY
SET REPORT MAX LINE LENGTH
SET REPORT SIG CHARACTERS

PANELS

Drawing Panels

BEGIN PANEL BOUNDARY
END PANEL
SELECT FILL PATTERN

PERIPHERAL DEVICES

Two-Port Peripheral Interface

MAP INDEX TO PEN
PLOT
PORT ASSIGN
REPORT PORT STATUS
SET PORT BAUD RATE
SET PORT EOF STRING
SET PORT FLAGGING MODE
SET PORT PARITY
SET PORT STOP BITS

PIXEL OPERATIONS

Data Transfer

PIXEL COPY
RASTER WRITE
RECTANGLE FILL
RUNLENGTH WRITE

Initialization

BEGIN PIXEL OPERATIONS
SET PIXEL BEAM POSITION
SET PIXEL VIEWPORT

SEGMENTS

Defining Segments

BEGIN HIGHER SEGMENT
BEGIN LOWER SEGMENT
BEGIN NEW SEGMENT
BEGIN SEGMENT
DELETE SEGMENT
END SEGMENT
INCLUDE COPY OF SEGMENT
RENAME SEGMENT
SET PIVOT POINT

Displaying Segments

SET FIXUP LEVEL

Reporting Segment Settings to the Host

REPORT SEGMENT STATUS

Segment Classes

SET CURRENT MATCHING CLASS
SET SEGMENT CLASS

Setting Segment Attributes

SET SEGMENT CLASS
SET SEGMENT DETECTABILITY
SET SEGMENT DISPLAY PRIORITY
SET SEGMENT HIGHLIGHTING
SET SEGMENT IMAGE TRANSFORM
SET SEGMENT POSITION
SET SEGMENT VISIBILITY
SET SEGMENT WRITING MODE

SURFACES

SET BACKGROUND COLOR
SET BACKGROUND INDICES
SET SURFACE COLOR MAP
SET SURFACE DEFINITIONS
SET SURFACE PRIORITIES
SET SURFACE VISIBILITY

Section 5

4100-STYLE PARAMETER TYPES, COMMANDS, AND REPORTS

INTRODUCTION

This section contains detailed descriptions of the terminal's 4100-style parameter types, commands, and reports. The explanations are for both the terminal's Host syntax and Setup mode syntax.

The terminal executes 4100-style commands when it is in Tek mode, which you specify with the SELECT CODE command.

The first part of this section includes descriptions of parameter types associated with 4100-style commands. The parameter types are explained in alphabetical order. Commands issued from the host use special encoding for parameter values. Setup mode commands use simple English words or numbers as parameters.

The second part of this section contains explanations of 4100-style commands and of reports associated with certain commands. Most commands have two forms: one used for sending the command from the host, and the other used locally in the terminal's Setup mode. Since the host form is in a cryptic opcode format, it is given a *descriptive* command name to clarify what the command does. The associated Setup mode command accomplishes the same function locally; it has an English-style name that identifies the command's purpose. The command explanations are given alphabetically according to the descriptive command name.

4100-STYLE PARAMETER TYPES

The following explanations document the various types of values used for the 4100-style parameters.

Character Array Parameters in Host Syntax

The character array parameter type allows you to send a list of characters as one parameter. This parameter type occurs only in the host syntax form of a command.

The first character in the array is an encoded integer indicating the length of the array. The integer is followed by the ASCII characters in the array.

The following example shows how the phrase **PRESS RETURN KEY** would be sent as a character array:

A0PRESS RETURN KEY

Character Parameters in Setup Syntax

The Setup mode form of commands have three kinds of character parameters: *small integer, string,* and *character*.

The Small Integer Parameter. *Small integer* parameters range from 0 through 127, which is the range of ASCII decimal equivalents for characters $^{N}{}_{U}$ through $^{D}{}_{T}$. When you specify a *small integer* parameter, you can use either the actual character or its ADE value.

The String Parameter. Some commands use an undelimited *string*. The word "undelimited" means that the parameter does not require special characters to distinguish it from other character strings. A *string* parameter requires that you specify actual characters rather than ADE values, and that you not use a comma or a space in the string.

The Character Parameter. The *character* parameter type corresponds to the printing characters listed on a standard ASCII chart. A *character* parameter has a range of $^{S}{}_{P}$ to ~ (tilde); it has an ASCII decimal equivalent (ADE) range of 32 to 126. When you specify a *character* parameter, you can use either the actual character or its ADE value.

All three of these parameter types can be used as delimited parameters. When parameters are delimited, the first and last characters have a special meaning to distinguish that group of characters from others. A delimited parameter can only include ASCII characters, not ADE values.

For example, in the Setup syntax of the SET EOF STRING command, you specify up to ten *small integer* characters to indicate the end-of-file marker. The delimiter can be any character (except a comma, a space, or an edit character), and is always the first character after the space following the command name. The delimiter is followed by up to ten alphanumeric characters and that group of characters is terminated by the same delimiter. The following command illustrates a delimited character parameter:

EOFSTRING 'THE END'

Key Specifiers in Setup Syntax

The *key specifier* parameter type is used in the DEFINE MACRO and DEFINE NONVOLATILE MACRO commands.

See Table 5-2 (under DEFINE MACRO) for the ASCII decimal equivalents (ADE) and macro numbers referred to in the following paragraphs.

Key specifiers for keys labeled with ASCII characters can be the ASCII character itself or a two- or three-digit ADE value or a macro integer number.

Function keys are represented by macro numbers or by a two-character abbreviation. The abbreviations F1 through F8 designate the unshifted versions of the function keys; S1 through S8 specify the shifted versions of those keys. The Ctrl and Shift-Ctrl versions of those keys are specified by macro numbers only.

The numeric keypad, special function keys, and joydisk are specified by macro numbers.

Keywords in Setup Syntax

The *keyword* parameter type is used in Setup mode only. Keyword parameters specify what action you want a command to perform. When you use a keyword, you can spell out the entire word or just as many characters as necessary to distinguish that keyword from other keywords associated with the command.

Integer Parameters in Host Syntax

In Host syntax, integers are encoded as a series of one, two, or three characters. For example, Figure 5-1 illustrates how that the integer −2413 is encoded as:

BV-

First, the decimal numeral must be translated into binary. Then the digits of the binary numeral are encoded (see Figure 5-1).

Hi-I: There may be one or two Hi-I characters, depending on the value of the integer. The Hi-I characters are:

1 d d d d d d

The first bit is always set to 1; the next six bits are the next six most significant bits in the binary numeral. In the example in Figure 5-1, the Hi-I character is:

1 0 0 0 0 1 0

This corresponds to the **B** character on an ASCII chart.

The second Hi-I character is:

1 0 1 0 1 1 0

This corresponds to the **V** character on an ASCII chart.

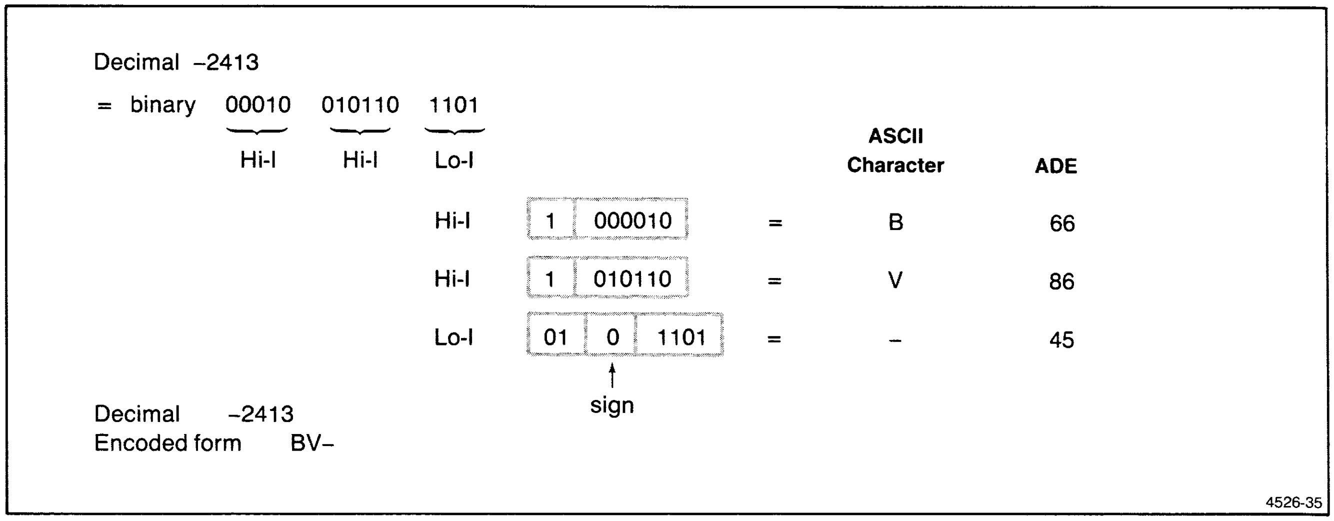


Figure 5-1. How Integers Are Encoded.

Lo-I: The Lo-I character may be the only character. It contains the sign bit and the four least-significant bits of the integer:

```
0 1 s d d d d
```

The first two bits are always set to 0 and 1; the third bit represents the sign of the integer. A negative integer is represented by a 0; a positive integer is represented by a 1. The next four characters are the least significant bits of the integer.

In the example in Figure 5-1, Lo-I would be:

```
0 1 0 1 1 0 1
```

This corresponds to the - (hyphen) character on an ASCII chart.

Integer Parameters in Setup Syntax

Whenever you use an integer parameter in setup syntax, simply enter the value. For example, to set both the transmit and receive rate to 2400 using the **BAUDRATE** Setup command, enter:

BAUDRATE 2400 2400

Small Integer Parameters in Host Syntax

In some commands, the parameter type called for is a *small integer*. Like the small integers used in Setup syntax, the host small integers represent ADE values from 0 through 127. However, in Host syntax you must encode these values, just as you encode any other integer (as explained in the preceding paragraphs). This means that the host must send 0 for the character ^{N}U (ADE 0), B0 for the character ^{S}P (ADE 32), C0 for the character 0 (ADE 48), and so forth through G? for the character ^{D}T (ADE 127).

Integer Array Parameters in Host Syntax

Some commands take integer array parameters. Integer array parameters consist of sequences of integer parameters. The first integer tells how many items are in the array; subsequent integers represent individual array items. You encode each item in the array the same way you encode a single integer, as described under "Integer Parameters in Host Syntax."

For example, you would send the array of integers (1, 5, −1, 16) to the terminal as follows:

```
integer array: (1,5,−1,16) = integer:4 (the count of 4)
                             integer:1
                             integer:5
                             integer:−1
                             integer:16
                           = 415!A0
```

Integer Array Parameters in Setup Syntax

Like the Host syntax integer array, the Setup syntax integer array consists of a sequence of integer parameters. Unlike the host, there is no item count at the beginning of the array, and there is no need to encode the integers. Following the Setup command name and a space, you simply enter the value of each item in the array, separating each item with a space or a comma.

Some commands that have multiple parameters may use an integer array for one or more of those parameters. If you need to include more than one integer array parameter in a command issued in Setup syntax, you must enclose all but the last integer array in angle brackets. For instance,

```
SGMATCHINGCLASS <13 14> 3 4 5
```

contains two integer arrays. The first consists of integers 13 and 14, and the second consists of integers 3, 4, and 5.

Real Parameters in Host Syntax

A *real* number is a variable which may assume fractional values. Real numbers between −32767.0 and 32767.0 are sent to the terminal as real parameters. These consist of a pair of encoded integers (see "Integer Parameters in Host Syntax"). The first integer represents a number; the second integer represents the power of two by which that number is to be multiplied.

For example, the number 1.5 may be expressed as 3 times ½, or 3 times 2 to the power −1; hence

```
real:1.5 = integer:3   integer:−1
         = 3!
```

The number pi may be approximated as 25736/8192, or 25736 times 2 to the power −13; hence

```
real:pi = integer:25736   integer:−13
        = YH8-
```

Real Parameters in Setup Syntax

In Setup syntax, the operator enters real parameters from the keyboard by entering the number and the power of two as integers (see "Integer Parameters in Setup Syntax"). The examples just given for Host syntax can be entered in Setup syntax as follows:

real:1.5 = **3 –1**
and
real:pi = **25736 –13**

XY-Coordinates in Host Syntax

The host must encode each xy-coordinate into one to five ASCII characters. The valid range of xy-coordinates is 0 to 4095. The xy-coordinates (53,1000), for example, would be encoded and sent to the terminal as the following characters:

'az^{S}PM

Figure 5-2 illustrates how the encoding process works. First, the decimal numeral is translated into a 12-bit binary numeral. Then the digits of the binary numeral are encoded. In the encoding process, the bits are divided into *Hi, Lo,* and *Extra* bytes.

In the example in Figure 5-2, decimal 53 translates to binary:

00000 01101 01

Decimal 1000 translates to binary:

00111 11010 00

The digits are encoded and transmitted in the order in which the following explanations are given:

Hi-Y: Contains the five most significant bits of the binary numeral representing the y-coordinate of the xy-parameter. The seven-bit Hi-Y character (excluding the parity bit) has the following format:

0 1 y y y y y

The first two bits must be set to **0 1**, as shown. The characters "y y y y y" represent the five most significant bits of the y-coordinate. In the example in Figure 5-2, this byte is:

0 1 0 0 1 1 1

This corresponds to the apostrophe (') character.

You can omit the Hi-Y byte if the five high-order bits of the y-coordinate have not changed since the last xy-coordinate was sent.

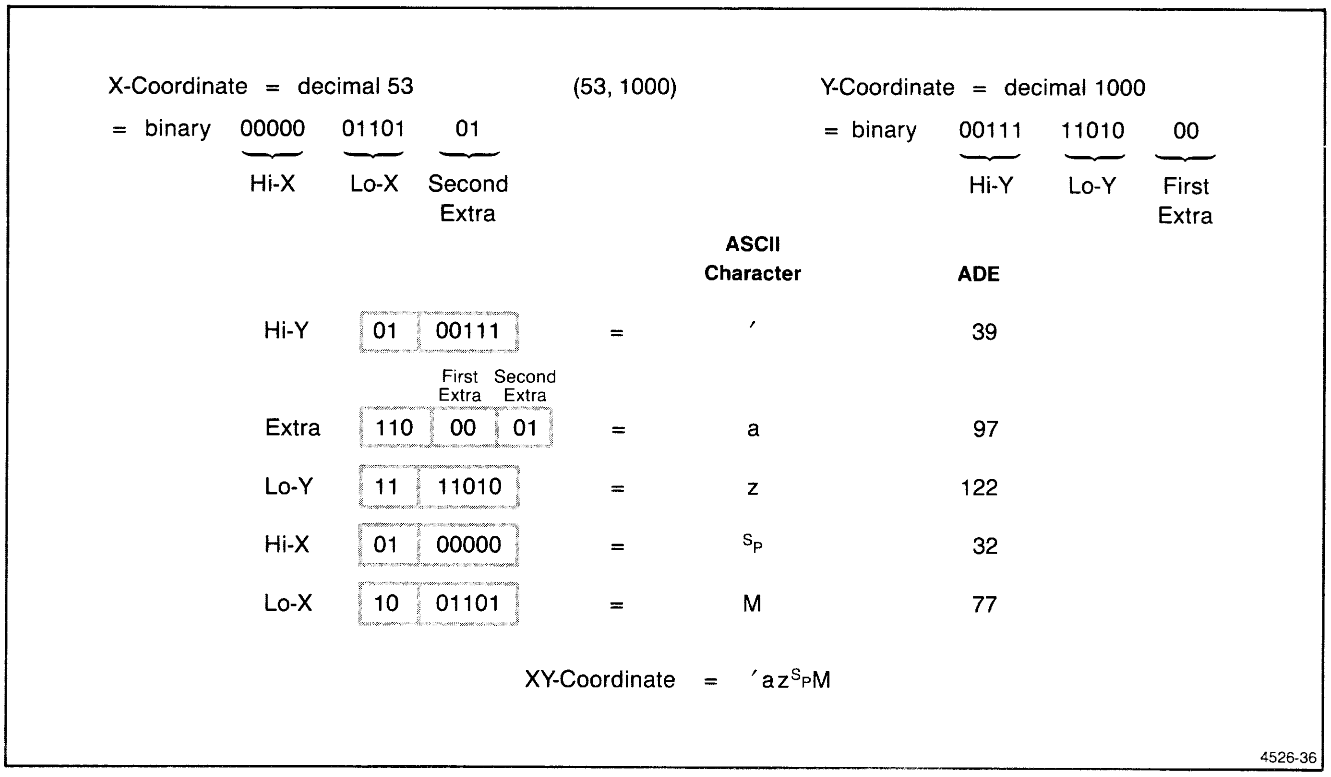


Figure 5-2. How XY-Coordinates Are Encoded.

Extra: Contains the two least significant bits of the x- and y-coordinates. The seven-bit ASCII character has the following format:

```
1 1 0 y y x x
```

The first three bits must be set to **1 1 0**, as shown. The characters "y y x x" represent the two least significant bits of the y- and x-coordinates, respectively. In the example in Figure 5-2 this byte is:

```
1 1 0 0 0 0 1
```

This corresponds to the letter **a** on the ASCII chart.

You can omit the Extra byte if you only want to send coordinates with 10 bits of precision or if the two least significant bits of the x- or y-coordinate have not changed since the last xy-coordinate. If you do send the Extra byte, you must follow it with a Lo-Y bite.

Lo-Y: Contains the five intermediate bits of the y-coordinate. The seven-bit ASCII character has the following format:

```
1 1 y y y y y
```

The first two bits must be set to **1 1**, as shown. The characters "y y y y y" represent the five intermediate bits of the y-coordinate. In the example in Figure 5-2, this byte is:

```
1 1 1 1 0 1 0
```

This corresponds to the letter **z** on the ASCII chart.

You can omit the Lo-Y byte if the following conditions are all true:

- These five bits have not changed since the last xy-coordinate.
- You are not sending the Extra byte.
- You are not sending the Hi-X byte.

Hi-X: Contains the five most significant bits of the x-coordinate of the xy-parameter. The seven-bit ASCII character has the following format:

```
0 1 x x x x x
```

The first two bits must be set to **0 1**, as shown. The characters "x x x x x" represent the five most significant bits of the x-coordinate. In the example in Figure 5-2 this byte is:

```
0 1 0 0 0 0 0
```

This corresponds to the ASCII SP (space) character.

You can omit the Hi-X character if the x-coordinate's most significant bits have not changed from the last xy-coordinate sent. If you do send the Hi-X character you must precede it with the Lo-Y byte.

Lo-X: Contains the five intermediate bits of the x-coordinate. The seven-bit ASCII character has the following format:

```
1 0 x x x x x
```

The first two bits must be set to **1 0**, as shown. The characters "x x x x x" represent the five intermediate bits of the x-coordinate. In the example in Figure 5-2, this byte is:

```
1 0 0 1 1 0 1
```

This corresponds to the letter **M** on the ASCII chart.

This character must always be sent, since it terminates the xy-parameter sequence.

NOTE

Since the Lo-Y and Extra bytes each have high-order bits of "1 1", DT (binary 1 1 1 1 1 1 1) could appear as a Lo-Y or Extra byte. This presents a problem for hosts that use DT as a filler character.

If your host uses DT as a filler, have the host do the following things:

- *Send the two-character string* **EC?** *instead of DT since the terminal recognizes that string as a synonym for DT.*
- *Send the terminal an IGNORE DELETES command to cause the terminal to ignore all DT characters sent from the host.*

Integer Report Parameters in Host Syntax

When the terminal sends integers to the host, it sends them encoded as three characters in an integer report. The way the terminal encodes integer reports is similar, but not identical, to the way the host encodes integer parameters (see "Integer Parameters in Host Syntax"). Integer report parameters have these characteristics:

- The terminal may include an EOL string, but sends it only if there is no other way to avoid exceeding the current maximum line length.
- Unlike integer parameters, integer reports always have three characters.
- The encoding scheme for an integer report's Lo-I character is identical to that of the integer parameter.
- The encoding scheme for an integer report's two Hi-I characters uses a different "offset" than for an integer parameter. In an integer parameter's Hi-I character, 64 is added to a six-bit binary numeral to form the ASCII decimal equivalent of the Hi-I character (this is the 1 seen at the beginning of the Hi-I's in Figure 5-1). In an integer report's Hi-I characters, 32 (rather than 64) is added to the six-bit binary numeral.

If, for example, we use the binary representation of –2413 shown in Figure 5-1, the first Hi-I character will be:

```
Six-bit binary       000010
Add offset of 32   + 0100000
                   = 0100010    = ASCII character "
```

The second Hi-I character will be:

```
Six-bit binary       010110
Add offset of 32   + 0100000
                   = 0110110    = ASCII character 6
```

The Lo-I character is encoded in the same way as the integer parameter and will be:

```
                     0101101    = ASCII character -
```

Thus, if the terminal sends the number –2413 to the host as an integer report, it encodes that number as **"6-**.

4100-STYLE COMMAND AND REPORT DESCRIPTIONS

This part of the section includes explanations of the terminal's 4100-style commands and reports. The commands are presented alphabetically according to their descriptive names. Figure 5-3 shows the format in which commands are explained.

COMMAND CONVENTIONS

There are four different kinds of 4100-style escape sequence commands:

- A single character
- The ^{E}C character followed by one character
- The ^{E}C character followed by two characters
- The ^{E}C character followed by two characters and one or more parameters

Figure 5-3 illustrates the general syntax conventions used in this section of the manual. Almost every command in this section has a Host syntax form; many commands have both a Host syntax form and an English-style Setup syntax form. Some commands have just a Setup syntax form.

NOTE

Parameters cannot be omitted from commands that you enter in Host syntax. If a parameter is omitted, the terminal interprets the next character sent by the host as the missing parameter. The results are unpredictable.

Parameters can be omitted from commands entered in Setup syntax from the keyboard. This is generally true if the parameter is the only one in the command or is the last of two or more parameters, but may not be true in all cases.

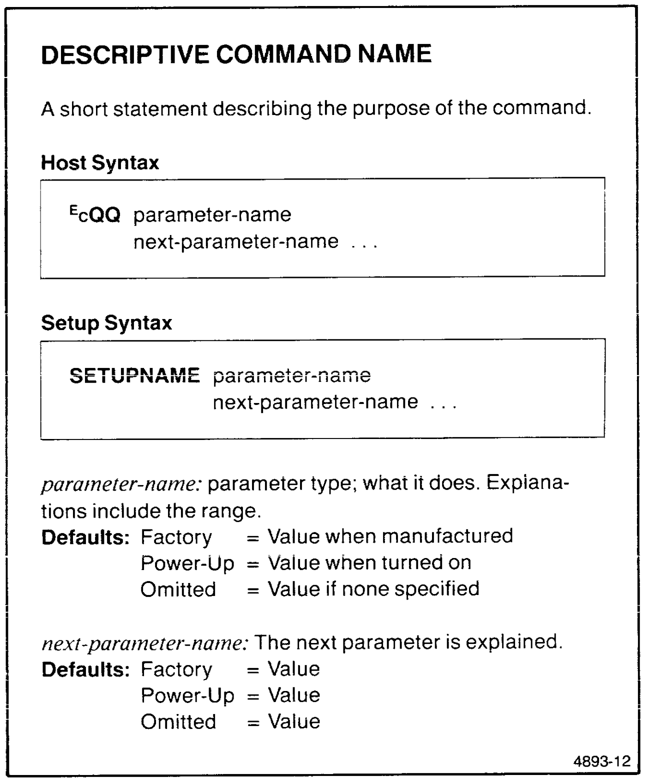
The following points summarize the formats illustrated in the syntax boxes in Figure 5-3.

- Characters shown in bold type are those you must enter exactly as shown.
- Parameter names are shown on separate lines to make the command syntax easier to read. However, when entering commands, follow these rules:
  - In Setup syntax, parameters should be on the same line. The first character after the command name must be a space; use one or more spaces or one comma to separate parameters.
  - In Host syntax, parameters should be on the same line with no spaces between any of the parameters, or between the $^{E}{}_{C}$ and following letters or parameters.
- Multiword items are joined by hyphens. For example, *parameter-name* and *next-parameter-name* are single items described by two or more words.
- Three periods following a parameter (. . .) indicate that it can be repeated.

As you can see in Figure 5-3, individual descriptions of each parameter follow the syntax boxes. Included are the parameter type, range of valid values, and default values. Any additional discussion of a parameter follows these parameter descriptions. The three default types have the following meanings:

- Factory — The value assigned a parameter when the terminal is shipped from Tektronix; parameters can be restored to this value by issuing the FACTORY command or running the Extended Self-Test program.
- Power-Up — The value assigned a parameter when the terminal is first turned on. The term "Saved in memory" indicates that the parameter takes on the value saved in the terminal's nonvolatile memory.
- Omitted — The value assigned a parameter if the command is used and no value is specified for the parameter. This is true only in Setup syntax; parameters cannot be omitted in Host syntax.

Any additional discussion or explanation of either a parameter or the command in general follows the parameter descriptions.



DESCRIPTIVE COMMAND NAME

A short statement describing the purpose of the command.

Host Syntax

$^{E}{}_{C}$**QQ** parameter-name
next-parameter-name . . .

Setup Syntax

SETUPNAME parameter-name
next-parameter-name . . .

parameter-name: parameter type; what it does. Explanations include the range.
Defaults: Factory = Value when manufactured
Power-Up = Value when turned on
Omitted = Value if none specified

next-parameter-name: The next parameter is explained.
Defaults: Factory = Value
Power-Up = Value
Omitted = Value

4893-12

Figure 5-3. Command Description Format for 4100-Style Commands.

BEGIN GRAPHTEXT CHARACTER

Starts the definition of a graphtext character in the specified graphtext font.

Host Syntax

```
EcST  font-number
      character-number
```

Setup Syntax

```
GTBEGIN  font-number
         character-number
```

font-number: integer; specifies the font number in which the character being defined resides. Must be in the range 0 through 32767.

character-number: integer; specifies the decimal equivalent of the character being defined. Must be in the range 32 through 126.

A graphtext character definition consists of a BEGIN GRAPHTEXT CHARACTER command, a number of MOVE and/or DRAW commands, and an END GRAPHTEXT CHARACTER command.

Before a graphtext character can be defined, a SET GRAPHTEXT FONT GRID must be saved for the specified font. The character being defined is displayed during its definition; to display the character after it has been defined, all the following conditions must be met:

- The current graphtext font is the same as the font named in the *font-number* parameter. (See the description of the SET GRAPHTEXT FONT command.)
- The current graphtext precision is "Stroke." (See the description of the SET GRAPHTEXT PRECISION command.)
- The specified character occurs within a GRAPHIC TEXT command's string parameter. (See the description of the GRAPHIC TEXT command.)

The character definition uses the current pivot point to position the font grid and to define the character origin. Displaying the character places its origin at the current graphics beam position.

Fonts 0, 1, 2, 3, 9, and 12 are predefined fonts for the ASCII, Swedish, German, United Kingdom, Danish/Norwegian, and French character sets, respectively. If you define a character in one of these predefined fonts, that user-defined character supercedes the corresponding predefined character. If you later delete the character definition with a DELETE GRAPHTEXT CHARACTER command, the character is restored to its predefined definition.

BEGIN HIGHER SEGMENT

Ends the current segment definition and begins a new segment that has an ID number one higher than the segment just ended.

Host Syntax

```
EcSN
```

Setup Syntax

```
SGUP
```

If no segment is currently being defined when this command is issued, the terminal detects an error and takes no action.

The pivot point and position of the new segment are set to the current beam position. The current Pick-ID is set to one.

The segment definition can be terminated by an END SEGMENT command, a BEGIN NEW SEGMENT command, a BEGIN LOWER SEGMENT command, or another BEGIN HIGHER SEGMENT command.

BEGIN LOWER SEGMENT

Ends the current segment definition and begins a new segment that has an ID number one lower than the segment just ended.

Host Syntax

```
EcSB
```

Setup Syntax

```
SGDOWN
```

If no segment is currently being defined when this command is issued, the terminal detects an error and takes no action.

The pivot point and position of the new segment are set to the current beam position. The current Pick-ID is set to 1.

The segment definition can be terminated by an END SEGMENT command, a BEGIN HIGHER SEGMENT command, a BEGIN NEW SEGMENT command, or another BEGIN LOWER SEGMENT command.

BEGIN NEW SEGMENT

Begins a new graphic segment definition.

Host Syntax

```
EcSE  segment-number
```

Setup Syntax

```
SGNEW  segment-number
```

segment-number: integer; specifies new segment's ID number. Valid segment ID numbers are 1 through 32767.
Defaults: Omitted = Error

The pivot point and position of the new segment are set to the current beam position. The pick-ID is set to one.

The segment definition can be terminated by an END SEGMENT command, a BEGIN HIGHER SEGMENT command, a BEGIN LOWER SEGMENT command, or another BEGIN NEW SEGMENT command.

BEGIN PANEL BOUNDARY

Starts a panel definition.

Host Syntax

```
EcLP  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.
Default: Omitted = (0,0)

draw-boundary: integer; choose one of the following:

0	Specifies that the fill pattern covers the panel boundary
1	Specifies boundary is displayed in the finished panel using the current line style and line index

Default: Omitted = 0

After issuing BEGIN PANEL BOUNDARY, you can define the boundary of the panel in either of two ways:

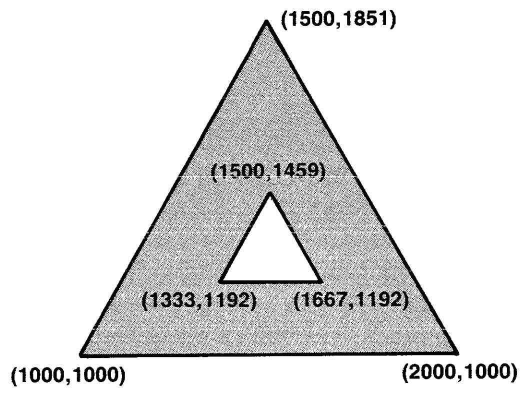
- Put the terminal in either Vector or Marker mode and then send the coordinates of the boundary line endpoints.
- Use MOVE and DRAW to define the boundary line.

You do not need to create the panel's last boundary line. When the terminal receives END PANEL, it closes the panel and fills it with the fill pattern specified in the most recent SELECT FILL PATTERN command.

You can define a panel when the terminal is in Marker mode. However, you cannot draw a marker during a panel definition when the terminal is in Marker mode.

If you define a panel while a segment is open (see the BEGIN SEGMENT command) the panel definition will be saved as part of the segment definition.

Multiple Panel Boundaries. You can create a panel with multiple boundaries as shown in Figure 5-4. To do this, send another BEGIN PANEL BOUNDARY when you want to start the second boundary (do not use END PANEL to close the first boundary). The second BEGIN PANEL BOUNDARY 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 defined by the multiple boundaries is filled.



Setup Command	Encoded Command From Host
BEGINPANEL 1000,1000,1	^{E}cLP'`z'Z1
DRAW 1500,1851	^{E}cLF. In+W
DRAW 2000,1000	^{E}cLF'`z/T
DRAW 1000,1000	^{E}cLF' `z'Z
BEGINPANEL 1333,1192,1	^{E}cLP)aj*M1
DRAW 1500,1459	^{E}cLF+I I+W
DRAW 1667, 1192	^{E}cLF)cj-@
END PANEL	^{E}cLE

4526-14A

Figure 5-4. Creating a Panel With Multiple Boundaries.

BEGIN PIXEL OPERATIONS

Sets parameters for use in subsequent RASTER WRITE, RUNLENGTH WRITE, RECTANGLE FILL, and PIXEL COPY commands.

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) their data. Must be chosen from the following:

−1	Specifies the super surface (all bit planes of all surfaces)
0	Specifies the current surface
1 — 4	Specifies a particular surface

Defaults: Factory = 1
Power-Up = 1
Omitted = 0

ALU-mode: integer; specifies arithmetic logic unit (ALU) writing mode. Valid values are 0, 7, 11, 12, or 15. Table 5-1 describes the function that each value selects.

Defaults: Factory = 11
Power-Up = 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
Power-Up = 6
Omitted = 0

NOTE

The terminal issues a Level 2 error if an undefined surface is specified (see SET SURFACE DEFINITIONS command).

Table 5-1

ALU VALUES

ALU Mode	Function	Application
0	no change	
7	A XOR B (exclusive OR)	Fast Erase mode. Writes an image that you can later completely remove by repeating the same operation. That is because A = (A XOR B) XOR B.
11	B	Replace mode. Replaces the existing image with the new pixels. This is the default *ALU-mode.*
12	A AND B	Performs a logical AND function on the pixel color indices and displays the results.
15	A OR B	Writes the new image on top of the existing image. Pixel values of 0 in the command string do not affect raster memory.

BEGIN SEGMENT

Begins the definition of a new segment, and resets the current Pick-ID to 1.

Host Syntax

```
EcSO segment-number
```

Setup Syntax

```
SGOPEN segment-number
```

segment-number: integer; Specifies the segment identification number. Valid segment numbers are 1 through 32767.
Defaults: Omitted = Error

If you specify a segment number for a segment that already exists, or if you use an invalid segment number, the terminal generates and reports an error. If another segment, graphtext character, or panel boundary definition is currently open, the terminal generates an error and ignores the command.

The segment will be defined with its pivot point at the location specified by the most recent SET PIVOT POINT command. (See the description of the SET PIVOT POINT command for details.) If the segment's pivot point is to be different from that previously defined, it is important to issue an appropriate SET PIVOT POINT command before issuing the BEGIN SEGMENT command.

A segment is made visible in the currently selected view when the END SEGMENT command terminates the segment definition, unless a SET SEGMENT VISIBILITY command for segment –2 has determined that newly-created segments are to be invisible.

If you issue a SELECT VIEW command during a segment definition, the segment is made visible only in the currently selected view when the segment is closed. To make a segment visible in all views, select each view and issue a SET SEGMENT VISIBILITY command with *segment-number* –2 for each view. This makes all future segments visible in each view.

CANCEL

Cancels terminal operations and several terminal parameters and modes.

Host Syntax

```
EcKC
```

Setup Syntax

```
CANCEL
```

This command cancels all current terminal activity and resets to default values a number of terminal parameters and modes.

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 modes:
  - Vector mode
  - Marker mode
  - GIN mode
  - Bypass mode
  - Prompt mode
  - Snoopy mode
- Removes the terminal from GIN mode as if a DISABLE GIN command were received. (See ENABLE GIN and DISABLE GIN commands.)
- Unlocks the keyboard. (See LOCK KEYBOARD command.)
- Terminates any copy or hardcopy function currently in progress. (See COPY command.)
- Flushes input and output queues. (See SET QUEUE SIZE command.) Characters not yet sent to the host will be discarded and ignored.

CLEAR DIALOG SCROLL

Clears (erases) the dialog buffer.

Host Syntax

${}^{E}{}_{C}$**LZ**

Setup Syntax

CLEARDIALOG

After the dialog buffer is cleared, the cursor returns to the Home position (upper-left corner of dialog area).

Issuing CLEAR DIALOG SCROLL has the same effect as pressing the terminal's D Eras key.

COPY

Sends data from the host directly to an attached hard copy unit.

Host Syntax

${}^{E}{}_{C}$**JC** source
separator
destination

Setup Syntax

COPY HO: TO destination

source: character array; **HO:** (in uppercase or lowercase) is the only valid value.

separator: character array; must be the characters **TO** (in uppercase or lowercase) or an empty array (parameter omitted).

destination: character array (string in Setup syntax); must be one of the following:

Host	Setup	
HC:	HC:	The terminal's hardcopy port
P0:	P0:	Peripheral Port 0
P1:	P1:	Peripheral Port 1

When the terminal receives this command, it passes all data it receives from the host directly to the specified port. The host is responsible for sending data that is meaningful to the device attached to the port. The copy terminates when the terminal detects an end-of-file string or when the operator presses the Cancel key.

CRLF

Specifies whether a ${}^{C}{}_{R}$ character implies a ${}^{L}{}_{F}$.

Host Syntax

${}^{E}{}_{C}$**KR** crlf-mode

Setup Syntax

CRLF crlf-mode

crlf-mode: integer (keyword in Setup syntax); 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 (no)
Power-Up = Saved in memory
Omitted = 1 (yes)

When ${}^{C}{}_{R}$ *does not imply* ${}^{C}{}_{R}{}^{L}{}_{F}$ is set, a ${}^{C}{}_{R}$ character is displayed as a Carriage Return only, not as a Carriage Return + Line Feed combination. A ${}^{C}{}_{R}$ moves the cursor to the beginning of the *current* line, not the next line.

When ${}^{C}{}_{R}$ *implies* ${}^{C}{}_{R}{}^{L}{}_{F}$ is set, a ${}^{C}{}_{R}$ from the host or from the keyboard (if the terminal is in Local mode and the operator presses the Return key) is displayed as a Carriage Return + Line Feed combination. The cursor moves to the beginning of the *next* line on the display.

This setting affects only a ${}^{C}{}_{R}$ sent to the terminal screen. When you press the Return key, for example, the implied ${}^{L}{}_{F}$ character is not sent to the host.

DEFINE MACRO

Creates and deletes macros, which can be expanded on command from the host, at the keyboard, or both.

Host Syntax

```
EcKD macro-number
     macro-contents
```

Setup Syntax

```
DEFINE macro-number
       string
```

macro-number: integer; (key specifier or integer in Setup syntax); specifies the name of the macro being defined. Valid values in the range –150 through 32767. A value of –1 deletes all volatile macros.
Default: Omitted = 0

macro-contents: integer array; specifies ASCII decimal equivalent (ADE) integers that represent the characters defining the macro. (Host syntax only.)

string: delimited string; defines the macro. To use CR or the special editing characters in the macro definition, you must precede them with the *literal character*; see SET EDIT CHARS for information on the literal character. (Setup syntax only.)

After a macro is defined, you can expand it either with EXPAND MACRO (for any macro) or by pressing a key corresponding to the macro number (for macros numbered –150 through 143, excluding –1).

The following example shows how to define Macro 500 as "XYZ".

1. Get ADE values for X, Y, and Z.

Character	ADE
X	88
Y	89
Z	90

2. Convert the integers 500, 88, 89, and 90 to the terminal's integer format.

Integer	Converted Format
500	_4
88	E8
89	E9
90	E:

3. Issue this command:

   EcKD_43E8E9E:

The integer **3** indicates that three values follow. This convention is followed for all integer arrays.

Volatile and Nonvolatile Macros. The *Graphics Terminal* section explains the difference between volatile and nonvolatile macros. DEFINE MACRO defines and deletes only volatile macros. To define a nonvolatile macro, use DEFINE NONVOLATILE MACRO followed by SAVE NONVOLATILE PARAMETERS.

Deleting Macros. To delete a specific macro in Setup syntax, enter DEFINE MACRO but don't include *string* after the macro number. In Host syntax, indicate a length of 0 for *macro-contents.* The following example shows the command in Host syntax form that deletes macro 500 (500 is _4 in integer format):

EcKD_40

To delete all macros, specify macro number –1. The following Host syntax command deletes all macros (–1 is ! in integer format):

EcKD!0

Key Macros. Table 5-2 shows how integers from −150 to 143 (excluding −1) correspond to the terminal's keys. Macros in this range are called *key macros*. Note that the integer for most keys is the ADE of the character the key normally produces.

As shown in Table 5-2, each key is associated with up to four macros: unshifted, shifted, Ctrl, and Ctrl-shifted (in some cases two or more of the unshifted/shifted positions access the same macro).

Note that when you define a key in Setup syntax you can either enter the key's ASCII decimal equivalent or just press the key (provided this key normally produces an ASCII character). Enter the macro's definition as a delimited string rather than an array of ADE integers. For example, to define the Q key to send the characters QUIT, enter:

DEFINE Q "QUIT"

Key-Execute Character. A special character, called the *key-execute character*, can be included in a key macro to cause a macro to be used only at the terminal instead of being sent to the host. Enclose the string you want used locally between two occurrences of the key-execute character named in the last SET KEY EXECUTE CHARACTER.

If the characters between the key-execute characters form a valid command, the terminal executes the command. Otherwise, the macro is displayed as if the operator typed it.

Each key-execute character the terminal encounters in a macro switches how key macros are used. 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."

Always include the second key-execute character in the macro. If you omit the second key-execute character, subsequent macros are expanded at the terminal, even if they are intended for the host. This would continue until the terminal expands a macro that includes the key-execute character.

Table 5-2

MACRO NUMBERS INVOKED BY KEYS[1]
(North American Keyboard)

Keys	Key Label	Un-Shift	Shift	Ctrl	Ctrl-Shift
Standard Keyboard and Function Keys	{ [	91	123	27	27
	! 1	49	33	49	33
	@ 2	50	64	50	0
	# 3	51	35	51	35
	$ 4	52	36	52	36
	% 5	53	37	53	37
	^ 6	54	94	54	30
	& 7	55	38	55	38
	* 8	56	42	56	42
	(9	57	40	57	40
	) 0	48	41	48	41
	_ -	45	95	45	31
	+ =	61	43	61	43
	}]	93	125	29	29
	Rub Out	127	-34	-35	-36
	Esc	27	-37	-38	-39
	~ |	124	126	124	126
	Q	113	81	17	17
	W	119	87	23	23
	E	101	69	5	5
	R	114	82	18	18
	T	116	84	20	20
	Y	121	89	25	25
	U	117	85	21	21
	I	105	73	9	9
	O	111	79	15	15

[1] See the appendices for macro numbers on other keyboards.

Table 5-2 (cont)

MACRO NUMBERS INVOKED BY KEYS (North American Keyboard)

Keys	Key Label	Un-Shift	Shift	Ctrl	Ctrl-Shift
Standard Keyboard and Function Keys (cont)	P	112	80	16	16
	` \	92	96	28	28
	Back Space	8	–40	–41	–42
	Line Feed	10	–43	–44	–45
	Tab	9	–46	–47	–48
	A	97	65	1	1
	S	115	83	19	19
	D	100	68	4	4
	F	102	70	6	6
	G	103	71	7	7
	H	104	72	8	8
	J	106	74	10	10
	K	107	75	11	11
	L	108	76	12	12
	: ;	59	58	59	58
	" '	39	34	39	34
	Return	13	–49	–50	–51
	Z	122	90	26	26
	X	120	88	24	24
	C	99	67	3	3
	V	118	86	22	22
	B	98	66	2	2
	N	110	78	14	14
	M	109	77	13	13
	< ,	44	60	44	60
	> .	46	62	46	62
	? /	47	63	47	63
	Space Bar	32	–52	–53	–54
	F1	128	136	–2	–10

Table 5-2 (cont)

MACRO NUMBERS INVOKED BY KEYS (North American Keyboard)

Keys	Key Label	Un-Shift	Shift	Ctrl	Ctrl-Shift
Standard Keyboard and Function Keys (cont)	F2	129	137	–3	–11
	F3	130	138	–4	–12
	F4	131	139	–5	–13
	F5	132	140	–6	–14
	F6	133	141	–7	–15
	F7	134	142	–8	–16
	F8	135	143	–9	–17
	G Eras Dialog	–111	–117	–123	–129
	Cancel Setup	–112	–118	–124	–130
	D Copy S Copy	–113	–119	–125	–131
	Menu	–114	–120	–126	–132
	D Eras S Eras	–115	–121	–127	–133
	Break	–116	–122	–128	–134
Numeric Keypad Keys	0	–55	–69	–83	–97
	1	–56	–70	–84	–98
	2	–57	–71	–85	–99
	3	–58	–72	–86	–100
	4	–59	–73	–87	–101
	5	–60	–74	–88	–102
	6	–61	–75	–89	–103
	7	–62	–76	–90	–104
	8	–63	–77	–91	–105
	9	–64	–78	–92	–106
	.	–65	–79	–93	–107
	,	–66	–80	–94	–108
	-	–67	–81	–95	–109
	Ent	–68	–82	–96	–110
Joydisk Positions	Right	–135	–139	–143	–147
	Up	–136	–140	–144	–148
	Left	–137	–141	–145	–149
	Down	–138	–142	–146	–150

DEFINE NONVOLATILE MACRO

Creates and deletes both the volatile and nonvolatile version of a macro.

Host Syntax

```
EcKO  macro-number
      macro-contents
```

Setup Syntax

```
NVDEFINE  macro-number
          string
```

macro-number: integer; (key specifier or integer in Setup mode); specifies the name of the macro being defined. Valid values in the range −150 through 32767. A value of −1 means "all macros," other macro numbers refer to individual macros.
Default: Omitted = 0

macro-contents: integer array; specifies ASCII decimal equivalent (ADE) integers that represent the characters defining the macro.
Default: Omitted = Empty array

string: delimited string; defines the macro. To use CR or the special editing characters in the macro definition, you must precede them with the literal character; See SET EDIT CHARS for information on the literal character. (Setup syntax only.)

This command has the same effect as the DEFINE MACRO command, except it defines both the volatile and nonvolatile version of a macro, rather than just the volatile version. The only distinction between the DEFINE MACRO and DEFINE NONVOLATILE MACRO commands is that after the terminal is reset or the power is turned off, macros defined with DEFINE NONVOLATILE MACRO will remain (if an NVSAVE command was issued before the terminal was reset or turned off).

NOTE

This command must be followed by SAVE NONVOLATILE PARAMETERS to actually save or delete a macro in nonvolatile memory.

To delete a single nonvolatile macro, use DEFINE NONVOLATILE MACRO specifying an array length of 0 for *macro-contents*, and then use SAVE NONVOLATILE PARAMETERS.

DELETE GRAPHTEXT CHARACTER

Deletes the named character of the specified user-defined graphtext font.

Host Syntax

```
EcSZ font-number
     character-number
```

Setup Syntax

```
GTDELETE font-number
         character-number
```

font-number: integer; specifies the font from which the character is deleted. Valid values are:

–1	All fonts
0 — 32767	Specifies a font

character-number: integer; specifies the decimal equivalent of a character within the specified font. Valid values are:

–1	All characters
32 — 126	Specifies a character

If you specify –1 for *font-number*, the command deletes the specified user-defined character in all fonts. If you specify –1 for *character-number*, the command deletes all user-defined characters in the specified font.

When a user-defined graphtext character is deleted, it reverts to the corresponding predefined character for that font. For most fonts, that is the same as the corresponding character in the standard ASCII font (Font 0). However, for Fonts 1, 2, 3, 9, or 12, this is the corresponding character in the predefined Swedish, German, United Kingdom, Danish/Norwegian, or French font.

Specifying a *character-number* of –1 not only deletes all the characters in the specified font, but also deletes the graphtext font grid for that font.

DELETE SEGMENT

Removes a segment from memory.

Host Syntax

```
EcSK segment-number
```

Setup Syntax

```
SGDELETE segment-number
```

segment-number: integer; specifies the number of the segment to be deleted. Valid values for *segment-number* are:

–3	All segments that match the current matching class
–1	All segments (except Segment 0)
1 — 32767	A specific segment

Defaults: Omitted = Error

If the segment is currently being defined, the segment definition is terminated, and then the segment is deleted.

The display is updated to the extent specified by the most recent SET FIXUP LEVEL command.

Segments cannot be protected against deletion. Before issuing the DELETE SEGMENT command, take care that the command is really needed and that the segment number being deleted is correct.

If all segments and all views are to be deleted, it is faster to delete views first, and then delete segments. It is also faster to set the fixup level to 0, delete the segments, renew the view, and then restore the original fixup level.

DELETE VIEW

Deletes a specified view.

Host Syntax

```
EcRK view-number
```

Setup Syntax

```
VDELETE view-number
```

view-number: integer; specifies the view to be deleted. Valid values for *view-number* are:

-1	All views
0	The current view
1 — 64	A specific view

Defaults: Omitted = 0

If the current view is deleted, the next view higher (or the lowest remaining view if none higher exist) becomes the current view. If there are no other views (after *view-number* = -1), then View 1 (the default view) is selected and initialized. (See the SELECT VIEW command for details about the default view.)

If you want to delete all segments and all views, it is faster to first delete views and then delete segments.

DISABLE GIN

Disables the GIN function on a named device.

Host Syntax

```
EcID device-function-code
```

Setup Syntax

```
GINDISABLE device-function-code
```

device-function-code: integer; disables the specified GIN device and it's function. See Table 5-6 under the ENABLE GIN command for valid codes.

This command disables the GIN function on the named device. If the RS-232 peripheral port has been programmed by the peripheral port system commands, the configuration in effect before the ENABLE GIN command will be restored when GIN is disabled.

You can disable all GIN devices by specifying -1 for *device-function-code*. If you specify a *device-function-code* for an already disabled device, the command is ignored. In a similar manner, if you specify a device that is not present, the command is ignored. When a device function combination is disabled, the terminal sends the rest of the GIN Report Sequence: the appropriate terminal signature character and EOM indicator.

DIM ENABLE

Enables or disables the automatic screen-dim feature.

Host Syntax

```
|    EcKG dim-code                                                |
```

Setup Syntax

```
| DIM dim-code                                                    |
```

dim-code: integer (keyword in Setup syntax); must be one of the following:

Host	Setup	
0	no	Disables automatic dim feature
1	yes	Dims screen after five minutes of no interaction.

Defaults: Factory = 1
Power-Up = Saved in memory
Omitted = 0

DRAW

Draws a vector from the current graphics position to the specified location.

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-coordinate.
Default: Omitted = (0,0)

DRAW has the same effect as sending the terminal an xy-coordinate when the terminal is in Vector mode. The terminal can execute DRAW when it is in any mode except Ansi or Bypass mode.

The terminal draws the vector in the current line style and line index. Use SET LINE STYLE to set the line style. The line index is set with SET LINE INDEX.

DRAW does not affect the terminal's mode. For example, if the terminal was in Alpha mode when it received the DRAW command, it stays in that mode.

See the section called *The Graphics Terminal* for examples of how to create a line.

DRAW MARKER

Draws a marker at a specified location.

Host Syntax

```
EcLH marker-position
```

Setup Syntax

```
MARKER marker-position
```

marker-position: xy-coordinate; indicates the point where the marker is drawn. Valid range of values is 0 through 4095 for both the x- and y-coordinate.
Default: Omitted = (0,0)

DRAW MARKER has the same effect as sending the terminal an xy-coordinate when the terminal is in Marker mode. The terminal can be in any mode except Ansi or Bypass mode when it executes DRAW MARKER.

The marker specified by the most recent SET MARKER TYPE is drawn in the current line index; SET LINE INDEX sets the line index.

DRAW MARKER does not affect the terminal's mode. For example, if the terminal is in Vector mode when it receives the DRAW MARKER command, it stays in that mode.

The writing mode specified by the SET GRAPHICS AREA WRITING MODE command affects the way that a marker is drawn on the screen. See the SET GRAPHICS AREA WRITING MODE command for details about how the writing modes operate.

ENABLE DIALOG AREA

Enables or disables the dialog area.

Host Syntax

```
EcKA  enable-mode
```

Setup Syntax

```
DAENABLE  enable-mode
```

enable-mode: integer (keyword in Setup mode); one of the following:

Host	Setup	
0	no	Disables the dialog area
1	yes	Enables the dialog area

Defaults: Factory = 1 (yes)
Power-Up = Saved in memory
Omitted = 1 (yes)

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 made visible; see SET DIALOG AREA VISIBILITY.)

When the dialog area is disabled and, the terminal is in Tek mode, the terminal emulates Tektronix 4010 Series terminals, which do not have a dialog area. The terminal displays alphatext at the current position in the graphics area.

In Ansi mode, the terminal automatically directs text to the dialog area, regardless of whether or not the dialog area is enabled.

Table 5-3 summarizes the effects of enabling and disabling the dialog area.

Table 5-3

EFFECTS OF ENABLE DIALOG AREA

Feature	Dialog Area Disabled	Dialog Area Enabled
Destination of Alphatext	Current position in terminal space	Current alpha cursor position in dialog area
G Eras Key S Eras Key and PAGE	Erases the graphics area of the screen (S Eras also erases dialog area) Takes the terminal out of GIN mode Resets the terminal to Line Style 1 Sets current position to Home position (0,3071) Puts terminal in Alpha mode	Erases the graphics area (S Eras also erases dialog area)
CR Character	Puts terminal in Alpha mode Performs a carriage return action Resets the terminal Line Style to 1 Takes the terminal out of GIN mode	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 GIN

Enables the terminal for graphics input (GIN).

Host Syntax

```
EC IE  device-function-code
       number-of-GIN-events
```

Setup Syntax

```
GINENABLE  device-function-code
           number-of-GIN-events
```

device-function-code: integer; enables the specified GIN device and the function for which that device is enabled. Device-function codes may be calculated by entering D (device) and F (function), from Tables 5-4 and 5-5, into the equation 8D + F. However, Table 5-6 lists all valid device-function codes.
Default: Omitted = 0

number-of-GIN-events: integer; specifies the number of points whose xy-coordinate position will be reported in a GIN Report Sequence. This value decrements toward 0 for each GIN Report Sequence. When the decrement value passes 1, the GIN feature automatically disables. Specifying a value of 0, enables GIN for 65535 events.
Default: Omitted = 65536

Table 5-4

DEVICE CODES

Device Code	Host Program Samples[a]	Event Device
0	Joydisk	Keyboard
1	4957 Tablet at P0	Tablet Pen or Puck
2	4957 Tablet at P1	Tablet Pen or Puck
6	4957 in Delta mode at P0	Tablet Pen or Puck
7	4957 in Delta mode at P1	Tablet Pen or Puck

[a] Only one device may be enabled on a given port at any time.

Table 5-5

FUNCTION CODES

Code	Function
0	Locator
1	Pick
2	Stroke

Table 5-6

VALID DEVICE-FUNCTION CODES

Device[a]	Function		
	Locate	Pick	Stroke
Joydisk	0	1	Not valid
Tablet Port 0	8	9	10
Tablet Port 1	16	17	18
Delta Mode Device P0	48	49	Not valid
Delta Mode Device P1	56	57	Not valid

[a] Only one device may be enabled on a given port at any time.

CAUTION

Undesired results may occur if the terminal is simultaneously enabled by graphics input commands ENABLE GIN and ENABLE 4010 GIN.

This command enables the terminal for graphics input from a keyboard joydisk, or external graphics tablet. The device code selection lets the terminal know how to interpret data from a specific input device. The function code selection prepares the terminal for the mode selections of Locator, Pick, or Stroke.

The GIN functions are:

Locator Function. When the Locator function is enabled, a graphics cursor appears on the screen to provide interactive position information for the operator. As the operator moves the input device (joydisk, tablet pen or puck) over the input surface, the graphics cursor on the screen moves along in exact correspondence with the input device's xy-coordinate location. Up to this point, no information has been sent to the host in the form of a GIN report sequence. This only occurs by signaling a GIN Event as follows:

● Joydisk	Press a keyboard key
● Pen	Press the pen point against the tablet surface
● Puck	Press appropriate button on puck

When the operator signals a GIN event, the GIN Report Sequence sent to the host contains the x- and y- coordinate for a particular location. (This is the location in terminal space of the graphics cursor. For further information, see the GIN Locator Report description, elsewhere in this section.)

NOTE

If the terminal is in Local mode then pressing a keyboard key cannot signal a GIN event, even though GIN may be enabled with the joydisk as the GIN device. The reason is that in Local mode characters typed on the keyboard are treated as if they came from the host rather than the keyboard.

In raster terminals, several points in terminal space can map to each pixel in raster memory space. Thus, when moving the GIN cursor slowly, it is possible to move the true cursor (whose position in terminal space is reported to the host) without producing any visible motion in the image of that cursor in raster memory space.

Likewise, if the window in terminal space is very small (as when the operator has "zoomed in" to see more detail), then adjacent points in terminal space may have images in the current viewport which are separated by some small distance. In that case, as the operator moves the cursor in terminal space, the image of that cursor moves in short steps from place to place in raster memory space.

Pick Function. When the Pick function is enabled, a graphics cursor appears on the the screen. Pick compares all detectable segments within the pick aperture, returning the segment number and Pick-ID of the highest priority such segment, together with the xy-coordinate of the graphics cursor. For further information, see the GIN Pick Report description, elsewhere in this section.

Stroke Function. When the Stroke function is enabled, the graphics cursor appears at a point on the screen which corresponds to the position on the tablet of the pen or tablet cursor. The operator begins a Stroke by pressing the tablet pen against the tablet, or pressing a button on the cursor tablet. The operator then moves the pen (or cursor) across the tablet surface and ends the Stroke by releasing pressure on the pen (or cursor button). During the Stroke, the terminal sends a continuous stream of x- and y-coordinates to the host. The quantity of reports sent to the host may be restricted by the SET GIN STROKE FILTERING command. Only the graphics tablet can be used with the stroke function. For more information, see the GIN STROKE REPORT command.

Delta Mode Device. The TEKTRONIX 4957 Graphics Tablet may also be enabled as a Delta mode (or mouse) device. When enabled in this mode, the GIN cursor position is not mapped directly from the position of the tablet puck, but is moved in the same direction as the puck is moved.

This mode of operation is very similar to thumbwheel or joydisk GIN. Large motions of the cursor can be accomplished with repeated small movements of the puck. The 4957's proximity zone is approximately 1.5 inches (4 cm) to keep the cursor from moving when you return the puck to the start of a movement it must be lifted out of proximity.

When Delta mode is first enabled, the GIN cursor is positioned at the last cursor position or at the position specified by the SET GIN START POINT command.

NOTE

The Stroke Function is not supported with the Delta Mode Device.

ENABLE KEY EXPANSION

Enables or disables key macros.

Host Syntax

```
EcKW macro-expansion-mode
```

Setup Syntax

```
KEYEXPAND macro-expansion-mode
```

macro-expansion-mode: integer; (keyword in Setup syntax); one of the following:

Host	Setup	
0	no	Disables key expansion
1	yes	Enables key expansion

Defaults: Factory = 1 (yes)
Power-Up = 1 (yes)
Omitted = 1 (yes)

With DEFINE MACRO, you can define a key, so that it produces a character (or characters) other than the character it normally produces. ENABLE KEY EXPANSION enables or disables alternate key definitions.

This command does not delete macros. All key macros remain in 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 EXPAND MACRO to expand any macros, including those associated with keys.

NOTE

While ENABLE KEY EXPANSION is disabled, all key macros temporarily revert to their factory default values.

ENABLE 4010 GIN

Enables the joydisk for one graphics input (GIN) Locator event.

Host Syntax

```
EcSB
```

This command is provided for compatibility with software written for earlier Tektronix terminals. It provides an abbreviated way of enabling the joydisk for one GIN (graphics input) Locator event. The report which the terminal sends in response to this GIN event is in the 4010 GIN Report format, rather than the GIN report sequence format used with the ENABLE GIN command.

CAUTION

Undersired results may occur if the terminal is simultaneously enabled by graphics input commands: ENABLE GIN and ENABLE 4010 GIN.

When the terminal receives an ENABLE 4010 GIN command, it displays the graphics cursor assigned to *device-function-code* 0. (The default cursor is the crosshair cursor; however, this may be changed with the SET GIN CURSOR command.)

The operator may then position the cursor by moving the joydisk. (The possible cursor positions are determined by the most recent SET GIN GRIDDING command for *device-function-code* 0 — joydisk device, Locator function.)

When the cursor is at the desired location, the operator presses any of the ASCII keys on the keyboard; this signals a GIN event. In response to this GIN event, the terminal sends a 4010 GIN Report. That report tells the host (a) which key the operator pressed, and (b) the location of the crosshair cursor in the terminal space.

When 4010 GIN is enabled, pressing a key that has a key macro defined for it will cause an event for the first character of the macro which would normally be sent to the host. The remainder of the macro is expanded normally.

When the terminal sends a report to the host, it enters Bypass mode. (See ENTER BYPASS MODE.)

After sending the 4010 GIN Report, the terminal sets its graphics beam position to the position of the graphics cursor, and enters Alpha mode. This is done for compatibility with Tektronix 4010 Series terminals.

If the dialog area is not enabled, then receiving a ^{C}R character, or a PAGE command, cancels the effect of the ENABLE 4010 GIN command and places the terminal in Alpha mode. Pressing the PAGE key has the same effect as the PAGE command. For details, see the description of the ENABLE DIALOG AREA command.

If the dialog area is enabled, then the PAGE key, the PAGE command, and the ^{C}R character have no effect on graphics input; they do not cancel the effect of the ENABLE 4010 GIN command.

Terminal Settings for Emulating 4010 Series Graphics Input. To properly emulate a Tektronix 4010 Series terminal during graphics input, several of the terminal's settings must be set a certain way. The exact settings may vary from one computer installation to another. Table 5-7 lists settings which should work for most host computers.

Table 5-7

TERMINAL SETTINGS TO EMULATE 4010 SERIES GRAPHICS INPUT

Host Command	**Equivalent Setup Mode Command**
SET REPORT EOM FREQUENCY: 1 = **^{E}cIM1**	REOM 1
SET EOM CHARS: 0, 0 = **^{E}cNC00**	EOMCHARS
SET EOL STRING: ^{C}R = **^{E}cNT** **integer-array: 13** = **^{E}cNT1** =	EOLSTRING ^{C}R
SET BYPASS CANCEL CHAR = **^{E}cNU** **integer:** *bypass-cancel-char*	BYPASSCANCEL ^{C}R BYPASSCANCEL ^{N}U BYPASSCANCEL ^{L}F

Set the *bypass-cancel-char* to whatever is the last character that the host echoes in response to a report message from the terminal. If the host is not echoing characters sent from the terminal, set the *bypass-cancel-char* to ^{N}U. If the host echoes ^{C}R as ^{C}R^{L}F, set the *bypass-cancel-char* to ^{L}F. If the host echoes ^{C}R as exactly ^{C}R, set the *bypass-cancel-char* to ^{C}R.

4010 GIN Report

When the operator presses an alphanumeric 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 operator pressed and the location of the crosshair 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.

The characters in the report are:

1. The ASCII character corresponding to the key the operator pressed.
2. The Hi-X character — the five most significant bits of x's binary value preceded by 01.
3. The Lo-X character — the five least significant bits of x's binary value preceded by 01.
4. The Hi-Y character — the five most significant bits of y's binary value preceded by 01.
5. The Lo-Y character — the five least significant bits of y's binary value preceded by 01.

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. After a host program converts reported coordinates by multiplying the values by 4, the derived values can differ from the true values by up to three units.

Example

If the operator positions the crosshair cursor at the point (50,1000) and then presses the lowercase "a" key, the following report is sent (in 1024x1024 terminal space):

a^{S_P},':

Figure 5-5 and the following explanations show why these five characters are sent.

ASCII Character. The first character in the report is *a*, since that was the key the operator pressed.

Hi-X Character. The x-coordinate in this example is 50, which converts to binary 0000 0011 0010. The Hi-X character always has the prefix 01 and is followed by the five high order bits in the binary numeral. So in this case, it is:

0100000

This is the space (^{S_P}) character.

Lo-X Character. This character has the prefix 01 followed by the five least significant bits of the x-coordinate. This produces:

0101100

This is a comma (,).

Hi-Y Character. The y-coordinate in this example is 1000, which converts to binary 0011 1110 1000. The Hi-Y character always has the prefix 01 and is followed by the five high order bits in the binary numeral. In this case, it is:

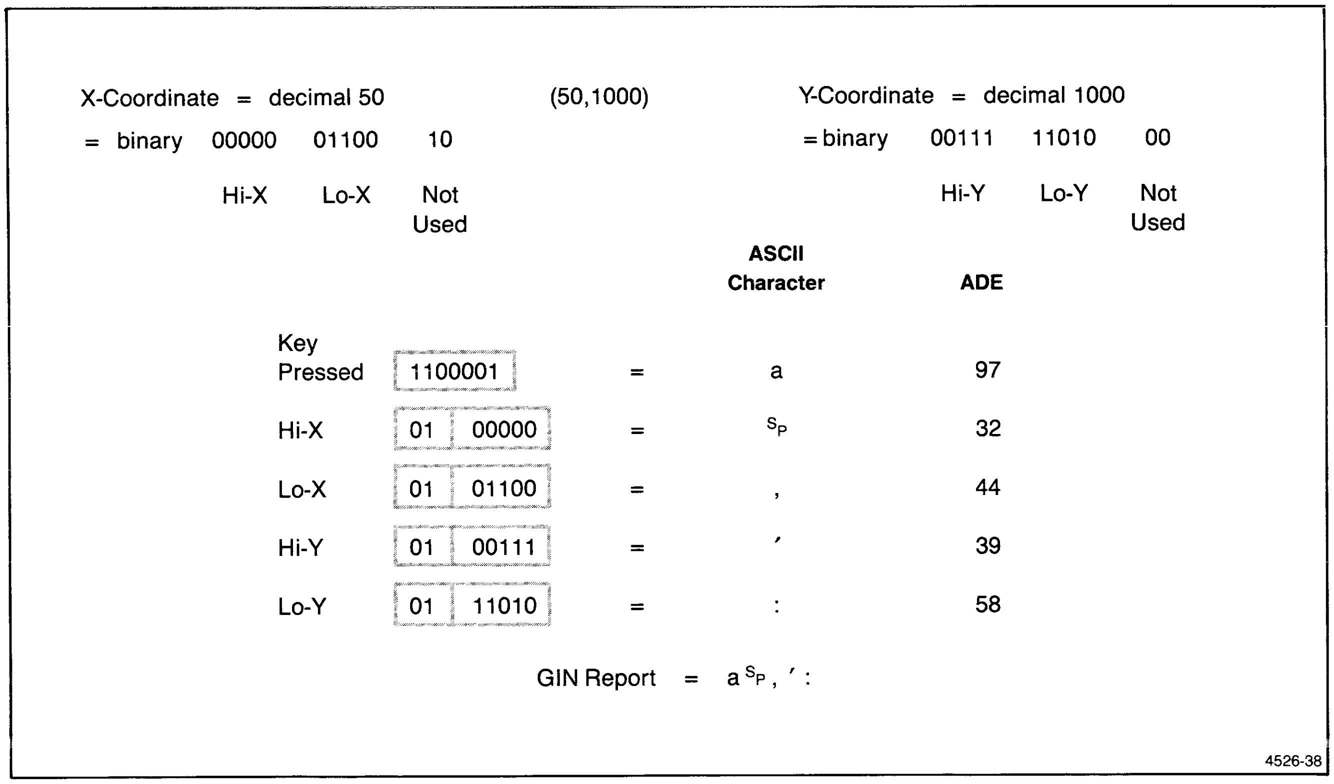
0100111

This corresponds to the apostrophe (') character.

Lo-Y Character. The Lo-Y character in the report consists of the prefix 01 followed by the five least significant bits in the binary numeral:

0111010

This corresponds to the colon (:) character.



			ASCII Character	ADE
Key Pressed	1100001	=	a	97
Hi-X	01 00000	=	^{S_P}	32
Lo-X	01 01100	=	,	44
Hi-Y	01 00111	=	'	39
Lo-Y	01 11010	=	:	58

GIN Report = a ^{S_P} , ' :

Figure 5-5. The GIN Cursor Position Report.

END GRAPHTEXT CHARACTER

Terminates the graphtext character currently being defined.

Host Syntax

```
EcSU
```

Setup Syntax

```
GTEND
```

END PANEL

Terminates a panel definition.

Host Syntax

```
EcLE
```

Setup Syntax

```
ENDPANEL
```

The panel boundary is closed, the panel is filled, and the current graphics beam position is set to the panel's first boundary point.

END SEGMENT

Terminates the segment currently being defined.

Host Syntax

```
EcSC
```

Setup Syntax

```
SGCLOSE
```

If a panel is currently being defined within the segment, that panel definition is also terminated, as if an END PANEL command had been received.

When the END SEGMENT command is issued, the segment that it terminates becomes visible in the current view (unless a SET SEGMENT VISIBILITY command for Segment −2 has specified that newly created segments are to be invisible).

ENTER ALPHA MODE

Puts the terminal in Alpha mode.

Host Syntax

```
Us
```

When the terminal is in Alpha mode, it interprets and displays ASCII characters as text. Alpha mode is the terminal's power-up default. Text is sent to the dialog buffer if the dialog area is enabled. If the dialog area is enabled and visible, text is also displayed in the dialog area as it is received. The Us character has an ASCII decimal equivalent (ADE) value of 31.

The terminal exits Alpha mode when it receives an ENTER VECTOR MODE or ENTER MARKER MODE command.

ENTER BYPASS MODE

Causes the terminal to enter Bypass mode.

Host Syntax

$^{E}_{C}{}^{C}_{N}$

This command causes the terminal to enter Bypass 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 leaves Bypass mode and discards the bypass-cancel character.

The terminal automatically enters Bypass mode when it sends the first character of a terminal-generated Tek mode message 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. For example:

1. Host sends $^{E}_{C}$**IQLI** (How many lines in dialog area?).
2. Terminal enters Bypass mode.
3. Terminal sends report, terminated by the EOM character.
4. Host echoes report to terminal. Report is ignored by terminal.
5. Host sends $^{L}_{F}$ (the bypass-cancel character).
6. Terminal cancels Bypass mode.
7. Host sends more data, which is processed by the terminal.

The terminal does NOT enter Bypass mode to send Ansi mode reports, since these reports do not include end-of-line strings.

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 of Bypass mode in the *Communications* section for more on Bypass mode. See also the SET BYPASS CANCEL CHARACTER command description later in this section.

ENTER MARKER MODE

Puts the terminal in Marker mode.

Host Syntax

$^{F}_{S}$

When the terminal is in Marker mode, it interprets ASCII characters as xy-coordinates and draws markers at the specified coordinates. (SET MARKER TYPE specifies the kind of marker the terminal draws.) The $^{F}_{S}$ character has an ASCII decimal equivalent (ADE) value of 28.

See "XY Coordinates in Host Syntax" at the beginning of this section for details on how to send those xy-coordinate parameters.

The terminal cannot go directly from Marker mode to Vector mode; it must first be placed in Alpha mode, then in Vector mode.

The use of Marker mode is explained and illustrated in *The Graphics Terminal* section.

ENTER VECTOR MODE

Puts the terminal in Vector mode.

Host Syntax

$^{G}_{S}$

When the terminal is in Vector mode, it interprets ASCII characters as xy-coordinates. These coordinates specify a point to which a vector is to be drawn or a point to which the graphics beam is to be moved. The cursor *moves* to the first xy-coordinate specified; it *draws* a vector to subsequent points. The $^{G}_{S}$ character has an ADE value of 29.

To draw rather than move when specifying the first coordinate after entering Vector mode, include the $^{B}_{L}$ character immediately after the $^{G}_{S}$ character.

See "XY-Coordinates in Host Syntax" at the beginning of this section for details on how to send those xy-coordinate parameters.

EXPAND MACRO

Expands the specified macro.

Host Syntax

```
EcKX  macro-number
```

Setup Syntax

```
EXPAND  macro-number
```

macro-number: integer; indicates the macro to expand. Valid range is −150 through −2 and 0 through 32767. When a −1 value (meaning all macros) is used with this command, a KX11 (Level 2) error is generated.
Defaults: Omitted = Unchanged

The EXPAND MACRO command causes the terminal to insert into its current input data stream the contents of a macro definition. It is a companion command to the DEFINE MACRO command, which defines a macro.

If the EXPAND MACRO command comes from the host computer, it is treated by the terminal as if the host computer had sent the contents of the macro which is being expanded.

Macros numbered from −150 through 143 may also be expanded by typing the corresponding key on the keyboard. The macro definition that is being expanded may contain other EXPAND MACRO commands. Commands may be nested this way to a nesting depth of at least 5.

Key Macros. Key macros are expanded with the EXPAND MACRO command or by pressing the key (or combination of keys) to which the macro corresponds.

The terminal treats any embedded EXPAND MACRO command as if the commands came from the host in either of these cases:

- An EXPAND MACRO command from the host triggered the expansion of the overall macro.
- The key associated with the macro is pressed while the terminal is in the Local mode with local echo enabled.

When a key macro is expanded by its corresponding keystroke, and Setup mode is disabled and the terminal is not in Local mode, the macro is sent to the host communication port as if it had been entered at the keyboard. This method only works if the key macro was expanded by a keystroke and the terminal is not in Setup mode. If Local mode or local echo are enabled, the macro's contents, including the key-execute character, is sent to the command processor, so that any EXPAND MACRO commands in the nested contents are treated as if they had come from the host.

When a key macro is expanded while Setup mode is enabled (either with a keystroke or with the EXPAND MACRO command), all characters within the macro-contents are treated as if they had been typed into the keyboard with Setup mode enabled, including EXPAND MACRO and other terminal commands.

EXPAND MACRO commands or keystrokes for macros that are already being expanded are ignored. Recursive macro expansions are not allowed.

With a GIN device enabled, pressing a key with a macro defined for it will cause a GIN event for each character in the macro which would normally be sent to the host (see the ENABLE GIN command).

Host Macros. Host macros are expanded with the EXPAND MACRO command.

EXPAND MACRO commands for macros that are already being expanded are ignored. Recursive macro expansions are not allowed.

If the EXPAND MACRO command is entered when the terminal is in SETUP mode, the contents of the specified macro are treated as if they had been typed on the keyboard while the terminal was in Setup mode.

FACTORY

Sets all parameters to their factory default values and takes the terminal out of Setup mode.

Setup Syntax

```
FACTORY
```

To set parameters permanently to the terminal's factory default, issue SAVE NONVOLATILE PARAMETERS command (**NVSAVE** in Setup) after issuing **FACTORY.**

GRAPHIC TEXT

Writes a string of graphtext starting at the current graphics position.

Host Syntax

```
EcLT text
```

Setup Syntax

```
GTEXT text
```

text: character array (delimited string in Setup syntax); indicates the characters to be displayed. Each character must have a decimal equivalent in the range 32 through 126 (SP through ~).
Defaults: Omitted = Empty string

The terminal draws the text string, starting at the current beam position, in the direction determined by the latest SET CHARACTER PATH command. After the string is drawn, the beam position is updated to a point also determined by the latest SET CHARACTER PATH command.

Stroke Precision Graphtext. If the most recent SET GRAPHTEXT PRECISION command specified stroke precision, the graphtext is drawn as a series of vectors. Its appearance is governed by the most recent SET TEXT INDEX, SET GRAPHTEXT SIZE, SET GRAPHTEXT FONT, SET GRAPHTEXT SLANT, and SET GRAPHTEXT ROTATION commands.

The text string is drawn in the current graphtext font, as determined by the most recent SET GRAPHTEXT FONT command. If no graphtext font has been selected, then Font 0 is used.

If the string is too long to fit in the 4096x4096 terminal space, the characters are clipped at the edge of terminal space. The current position is set to the edge of terminal space where the overflow occurred.

Graphtext cannot be included in a panel. Therefore, if you attempt to use GRAPHIC TEXT between BEGIN PANEL BOUNDARY and END PANEL commands, an error results and the panel is closed as if END PANEL had been executed.

String-Precision Graphtext. If the most recent SET GRAPHTEXT PRECISION command specified string-precision, the graphtext is displayed as if it were alphatext, except that it does not wrap at the right edge of the display. The appearance of string precision graphtext is governed by the SET TEXT INDEX, SET GRAPHTEXT SIZE, and SET GRAPHTEXT ROTATION commands.

The ANSI command SCS (SELECT CHARACTER SET) determines the character set used for graphtext. See the section titled *Screen Editing* for more information on character sets.

The mode (Overstrike mode or Replace mode) selected by the SET GRAPHIC AREA WRITING MODE command affects the way that string precision graphtext is written. When in Replace mode, the terminal writes both the character *and* character background of each character cell to the graphics area. The character is set to the current text index, and the character background is set to the text background index specified by the most recent SET BACKGROUND INDICES command. When in Overstrike mode, only the character in each character cell is written. Stroke precision graphtext is not affected by Overstrike/Replace mode; it is always written as if the terminal were in Overstrike mode.

HARDCOPY

Causes an attached hardcopy unit to make a copy of the terminal's screen or dialog area.

Host Syntax

```
EcKH 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	Produces a positive hardcopy of the entire screen
3	Copies only the dialog area

If you use 0, 1, or 3, the copy will be a negative image of the display (white areas copy black, black areas copy white), which is the way a hardcopy normally appears and typically is most useful. If you prefer a positive hardcopy image, use *hardcopy-code* 2 in the command.

Using this command has the same effect as pressing the S Copy, Ctrl-S Copy, or D Copy keys (*hardcopy-codes* 0 and 1, 2, and 3, respectively). To copy only the graphics area, make the dialog area invisible and use 0 or 1.

HELP

Displays the escape sequence, Setup name, and parameter types for a command or cluster of commands.

Setup Syntax

```
HELP name
```

name: string; either a Setup mode command name or the name of a cluster of commands for which you want information. The cluster names are COMMUNICATIONS, GIN, HARDCOPY, and PIXELS.
Default: Omitted = All commands

IGNORE DELETES

Determines whether the terminal ignores the $^{D}{}_{T}$ (Delete) character.

Host Syntax

```
EcKI ignore-deletes-mode
```

Setup Syntax

```
IGNOREDEL ignore-deletes-mode
```

ignore-deletes-mode: integer (keyword in Setup syntax); must be one of the following:

Host	Setup	
0	no	Terminal does not ignore $^{D}{}_{T}$ characters.
1	yes	Terminal ignores $^{D}{}_{T}$ character.

Defaults: Factory = 0 (no)
Power-Up = Saved in memory
Omitted = 1 (yes)

Some computers use $^{D}{}_{T}$ characters (Delete or Rubout; ASCII decimal equivalent 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 $^{D}{}_{T}$ is a valid character in integer and xy-coordinate parameters, there are times when the terminal should process $^{D}{}_{T}$s 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 $^{D}{}_{T}$ character.

See the heading "Coping With $^{D}{}_{T}$ Filler Characters" in the *Communications* section for details on how to use this command.

INCLUDE COPY OF SEGMENT

Copies the designated segment into the segment currently being defined.

Host Syntax

```
EcLK segment-number
```

Setup Syntax

```
SGINCLUDE segment-number
```

segment-number: integer; specifies the number of the segment to be included. Valid values for *segment-number* are:

-3	All segments that match the current matching class
-1	All segments
1 — 32767	A specific segment

Defaults: Omitted = Error

When this command occurs in a segment definition, the terminal does the following:

1. Ends and fills a panel if one is being defined.
2. Saves the current segment attributes (graphics beam position, line style, line index, text index, marker type, and pick-ID).
3. Copies the designated segment (scaled, rotated, and positioned according to its current image transform parameters) into the segment being defined.
4. Restores the current segment attributes previously saved.
5. Displays the included segment in the current writing mode (that of the default segment) if no segment is currently being defined.

When the included segment is copied, the initial default pick-ID is not copied, but all explicitly set pick-IDs are copied. None of the included segment's attributes, such as highlighting or detectability, are carried over into the new segment.

LEARN / NVLEARN

Programs keys from the keyboard.

Setup Syntax

```
LEARN or NVLEARN
```

A key programmed with **LEARN** remains programmed only until the terminal is turned off. A key programmed with **NVLEARN** remains programmed until you reprogram it, even if the terminal is turned off. To program a key, first enter Setup mode, then type:

LEARN CR or **NVLEARN** CR

The terminal will respond with the message:

Press the key to be defined:

This key can be any key on the keyboard (except Cancel, Caps Lock, Ctrl, and Shift) including function keys, the Break key, a shifted space bar, etc. If you press an ASCII key, the terminal echoes the ASCII character on the screen. If you press a function key or one of the special keycodes, such as Shift-Return, the terminal echoes the decimal macro number. Then the terminal displays this message:

Enter definition. (F1 terminates definition, F2 deletes last character)

Now simply type in the ASCII definition (you don't need delimiters). If you make a mistake, use F2 to delete characters. Each ASCII character you enter is added to the definition until you press F1, which ends the definition and puts you back into Setup mode. If you press any non-ASCII key (except F1 and F2) while typing the definition, the terminal ignores that key-press and rings its bell.

You can cancel the definition before ending it by pressing the Cancel key.

If there is not enough memory available to store a definition when you issue the LEARN command, the terminal returns control to Setup mode after displaying this message:

Error: Not enough memory available.

If, however, memory fills up *while* you are writing the definition, the terminal rings its bell and ignores any other keys you press.

LFCR

Specifies whether an LF character implies a CR.

Host Syntax

```
EcKF lfcr-mode
```

Setup Syntax

```
LFCR lfcr-mode
```

lfcr-mode: integer (keyword in Setup syntax); must be one of the following:

Host	Setup	
0	no	LF does not imply CR
1	yes	LF implies CR

Defaults: Factory = 0 (no)
Power-Up = Saved in memory
Omitted = 1 (yes)

When *LF does not imply CR* is set, the LF character is displayed as a Line Feed only, not as a Line Feed + Carriage Return combination.

When *LF implies CR* is set, an LF character from the host or from the keyboard (if the terminal in Local mode and the operator presses the Line Feed key) is displayed as a Line Feed + 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.

LOCAL

Enters and exits Local mode.

Setup Syntax

```
LOCAL local-mode
```

local-mode: keyword:

yes	Terminal enters Local mode
no	Terminal exits Local mode

LOCK KEYBOARD

Disables or enables the keyboard.

Host Syntax

```
EcKL locking-mode
```

locking-mode: integer; specifies whether the keyboard is locked or unlocked.

0	Unlocks the keyboard
1	Locks the keyboard

Defaults: Factory = 0
Power-Up = 0
Omitted = 0

The LOCK KEYBOARD command lets the host computer disable the keyboard keys. 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 (as opposed to the host echoing bells) 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 host will not echo a bell. (This is useful at times when a host computer program cannot tolerate input from the operator.)

The keyboard can be unlocked by issuing a KEYBOARD LOCK command with a parameter of 0. It can also be unlocked by pressing the CANCEL or BREAK key, or issuing a CANCEL command.

LOCK VIEWING KEYS

Locks and unlocks the Zoom and Pan modes by disabling the F2 function *frame* key.

Host Syntax

```
EcRJ locking-mode
```

Setup Syntax

```
LOCKVIEWINGKEYS locking-mode
```

locking-mode: integer; (keyword in Setup syntax); one of the following:

Host	Setup	
0	no	Unlocks the viewing keys
1	yes	Locks the viewing keys

Defaults: Factory = 0
Power-Up = 0
Omitted = 0

LOCK VIEWING KEYS disables the "frame" key when the Menu key's menu is displayed. Function key F2 is disabled so that pressing it only causes the terminal's bell to sound. That way, the operator cannot put the terminal in Zoom or Pan mode.

The host computer can inquire whether a LOCK VIEWING KEYS command has been executed by issuing a Report Terminal Settings command for "RJ" opcode. The Terminal Setting Report sent in response to such a command not only tells the host whether the viewing keys are locked, but also reports whether the terminal is in Zoom or Pan sub-mode of Framing mode. For details, see the description of the Terminal Settings Report under REPORT TERMINAL SETTINGS.

MACRO STATUS

Displays the definition of one or all macros.

Setup Syntax

```
MACROSTATUS macro-number
```

macro-number: integer or key specifier; specifies the macro whose definition you want displayed. Integer must be in the range −150 through 32767. The integer −1 or the keyword **all** displays all macros.
Default: Omitted = All

MAP INDEX TO PEN

Assigns the specified color index to a particular plotter pen.

Host Syntax

```
EcPI port-identifier
     index
     pen-ID-number
```

Setup Syntax

```
PMAP port-identifier
     index
     pen-ID-number
```

port-identifier: string; must be **P0:** or **P1:**. This selects the port to which the specified index is to be mapped.

index: integer; must be in the range from −1 through 255. A value from 0 through 255 specifies a particular color index; a value of −1 specifies "all indices."

pen-ID-number: integer; must be in the range from 0 through *n*, where *n* is the maximum number of pens for the plotter attached to the given port. A value of 0 specifies "no pen"; an index assigned to Pen 0 will not be drawn when the display is plotted. Values of *n* are:

Plotter	*n*
4662	1
4662 with multiple pens	8
4663	2

This command assigns a specific color index to a specific plotter pen number at the specified peripheral port. When graphic data is drawn on a plotter attached to that port, all color indices assigned the same *pen-ID-number* will be drawn with the same plotter pen.

To draw all information in the same color, use an *index* value of −1. To omit information with a particular color index from a plotted image, give that index a *pen-ID-number* of 0.

MOVE

Sets the current 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: Omitted = (0,0)

This command does not change the display. It is analogous to lifting a pen from the paper in a drawing and moving it to a new location.

The section titled *The Graphics Terminal* gives examples of how to use this command with the DRAW command to create lines.

PAGE

Erases the screen (except the dialog area).

Host Syntax

```
ᴇcFF
```

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 portion of the screen and renews the current view (see the RENEW VIEW command).

If the dialog area is not enabled, the terminal erases the graphics portion of the screen, renews the current view, and:

- Resets the current line style to 0 (solid lines)
- Terminates 4010 GIN mode
- Returns the graphic beam to its home position (0,3071)
- Puts the terminal into Alpha mode
- Cancels a command in Host syntax being entered from the keyboard or the host (Setup commands are not affected)

PIXEL COPY

Copies pixels from one rectangular region to another.

Host Syntax

```
ᴇcRX 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. Must be one of the following:

−1	Specifies the super surface (all bit planes of all surfaces)
0	Specifies the current surface
1 — 4	Specifies a particular surface

Defaults: Omitted = 0

destination-lower-left-corner: xy-coordinate; specifies the lower-left corner of a rectangular region on the destination surface. The range of values for x is 0 through 639; for y, 0 through 511.
Defaults: Omitted = (0,0)

first-source-corner: xy-coordinate; specifies one corner of a rectangular region on the current pixel surface. The valid range of values for x is 0 through 639; for y, 0 through 511.
Defaults: Omitted = (0,0)

second-source-corner: xy-coordinate; specifies the corner opposite the first source corner in the source region. The valid range of values for x is 0 through 639; for y, 0 through 511.
Defaults: Omitted = (0,0)

The PIXEL COPY command 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 PIXEL COPY command uses the ALU mode specified in the most recent BEGIN PIXEL OPERATIONS command. Copying pixels to the same location on the pixel writing surface can only occur in XOR mode to erase the pixels, or in other ALU modes to form mirror images (see following "Source Corners" discussion). Using other ALU modes to copy (without mirroring) to the same location will do nothing, and will not generate an error.

If the destination surface has fewer bit planes than the current surface, the destination surface receives only the most significant bits of each pixel index when the pixel is copied. For instance, if the current surface has three bit planes and the destination surface has only one, a pixel index with a binary value of 100 is copied to the destination surface as 1.

You can copy pixels using the off-screen raster memory with y values from 480 to 511, but a Level 0 warning will be generated.

NOTE

The terminal issues a Level 2 error if an undefined surface is specified (see SET SURFACE DEFINITIONS command).

Source Corners. The two source corners need not be the lower-left and upper-right corners of the source region. However, if they are not, the pixels written to the destination region may form a mirror (or inverted) image of the picture formed by the pixels in the source region. The pixel at the first source corner is copied onto the pixel at the lower-left corner of the destination region. The pixel at the second source corner is copied onto the pixel at the upper-right corner of the destination region.

PLOT

Sends all visible segments to a specified output device; if device is a plotter, segments are drawn on the plotter.

Host Syntax

```
EcPL  separator
      output-specifier
```

Setup Syntax

```
PLOT TO  output-specifier
```

separator: string; must be the string **TO** (upper or lower case) or omitted.

output-specifier: string; specifies where information is to be sent by the PLOT command. Must be one of the following:

P0:	Specifies Peripheral Port 0
P1:	Specifies Peripheral Port 1

This command converts visible segments from the current view into escape-sequence commands and sends them to the specified output device. Only those segments — or parts of segments — within the window for the current view are sent to the output device. Segments that are clipped on the display at the boundaries of the current view will be clipped accordingly when they are sent to the output device. SET WINDOW and SET VIEWPORT commands are also sent, so that the output device knows the location of the view's window and viewport.

If the most recent PORT ASSIGN command has assigned a plotter protocol to that port, then the terminal sends escape sequences in a format that the specified plotter can interpret. Otherwise, the terminal sends 4110 Series escape-sequence commands to the destination device.

The terminal terminates the transmission with the current end-of-file string for that port, as set by the SET PORT EOF STRING or SET EOF STRING command.

PORT ASSIGN

Assigns a device protocol to a specified peripheral port.

Host Syntax

```
EcPA port-identifier
     protocol-identifier
```

Setup Syntax

```
PASSIGN port-identifier
        protocol-identifier
```

port-identifier: string; must be one of the following:

P0:	Specifies Peripheral Port 0
P1:	Specifies Peripheral Port 1

Defaults: Power-Up = Saved in memory
Omitted = Error

protocol-identifier: string; must be one of the following:

PPORT	Assigns a general purpose RS-232 protocol.
4662	Assigns a protocol for a 4662 Plotter.
4662/MP	Assigns a protocol for a 4662 Plotter equipped with Option 31 (multiple pens). This allows graphics with different color indices to be drawn with different plotter pens (see MAP INDEX TO PEN).
4663	Assigns a protocol for a 4663 Plotter.

Defaults: Power-Up = Saved in memory
Omitted = Error

This command assigns the specified device protocol to the specified port.

PPORT Protocol. This is a general purpose protocol. When this protocol is used, the terminal does not format data sent to the specified port. Whatever data is sent to the port should already be in a format that the attached device can understand.

Plotter Protocols. These protocols assume that the appropriate Tektronix plotter is connected to the port and ready to receive data. Graphics commands are translated into escape sequence commands that the specified plotter can understand. Each of these protocols uses the plotter's block mode communications protocol. If tablet GIN is enabled for a particular port and the host issues a PASSIGN command for that port, a type PA11 error will occur: Port is in use.

PROMPT MODE

Initiates or terminates Prompt mode.

Host Syntax

```
EcNM prompt-mode
```

Setup Syntax

```
PROMPTMODE prompt-mode
```

prompt-mode: integer (keyword parameter in Setup syntax); must be one of the following:

Host	Setup	
0	no	Terminates Prompt mode
1	yes	Initiates Prompt mode after the next end-of-message character or end-of-line string is sent
2		Initiates Prompt mode immediately (Host syntax only)

Defaults: Factory = 0 (no)
Power-Up = 0 (no)
Omitted = 1 (yes)

See Section 2, *Communications*, for an explanation of Prompt mode.

RASTER WRITE

Specifies individually the color indices of a specified number of pixels in the current pixel viewport starting at the current beam position.

Host Syntax

$^{E}_{C}$**RP** number-of-pixels color-index-codes

Setup Syntax

PXRASTERWRITE number-of-pixels color-index-codes

number-of-pixels: integer; specifies the number of pixels represented by the character array that follows this parameter. Must be in the range 0 through 65535.
Defaults: Omitted = Error

color-index-codes: character array; specifies in a packed format the color indices for the pixels specified by *number-of-pixels.* Each code is an ASCII character in the range $^{S}_{P}$ through ' (ADE 32 through 96).
Defaults: Omitted = 0

You can regard the data bits embedded within the code characters in the *color-index-codes* array as a sequential string of bits. The bits are grouped to form color indices for individual pixels according to the *bits-per-pixel* parameter in the most recent BEGIN PIXEL OPERATIONS command. Figures 5-6 and 5-7 show how to pack color indices into the *color-index-codes* parameter.

If *bits-per-pixel* is 3, then pack the color indices 0, 0, 2, 3, 2, 7 into a RASTER WRITE command as follows:

1. Express the color indices as three-bit binary numerals:

$$\begin{tikzcd} 0 \arrow[d] & 0 \arrow[d] & 2 \arrow[d] & 3 \arrow[d] & 2 \arrow[d] & 7 \arrow[d] \\ 000 & 000 & 010 & 011 & 010 & 111 \end{tikzcd}$$

2. Group the binary bits into six-bit groups:

$$\begin{tikzcd} \underbrace{000\quad 000} \arrow[d] & \underbrace{010\quad 011} \arrow[d] & \underbrace{010\quad 111} \arrow[d] \\ 000000 & 010011 & 010111 \end{tikzcd}$$

3. Add 32 (binary 100000) to these six-bit binary numerals to form seven-bit ASCII characters:

$$\begin{tikzcd} 0100000 \arrow[d] & 0110011 \arrow[d] & 0110111 \arrow[d] \\ {}^{S}_{P} & 3 & 7 \end{tikzcd}$$

4. 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 = $^{E}_{C}$**RP 6 3**$^{S}_{P}$**37**

4526-39

Figure 5-6. Packing Color Index Codes Using Three Bits per Pixel.

If you use 1 for the *bits-per-pixel* parameter in the BEGIN PIXEL OPERATIONS command, 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. If *bits-per-pixel* is 3, then two color indices fit into each code character, as shown in Figure 5-6.

If the *bits-per-pixel* parameter is set to 4, one-and-a-half color indices fit into each code character. That is, every pair of codes holds three color indices. Figure 5-7 shows the packing scheme.

If the *bits-per-pixel* parameter is 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 15 with a single ASCII character.

If the number of bits per pixel is greater than the number of bit planes on the current surface, the surface receives only the most significant bits of each color index sent. For instance, if the number of bits per pixel is 3 and the current surface has only two bit planes, a color index with a binary value of 110 is written to the current surface as 11. In the case of six bits per pixel, if Color Index 8 (binary 001000) is sent to a surface that has three bit planes, the bits 1000 are sent and written to the surface as 100.

The special code character ‘ (ADE 96), functions much like a $^{C}_{R}{}^{L}_{F}$ 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.

If *bits-per-pixel* is 4, then pack the color indices 0, 0, 2, 3, 12, 15 into a RASTER WRITE command as follows:

1. Express the color indices as four-bit binary numerals:

```
  0      0      2      3      12     15
  ↓      ↓      ↓      ↓      ↓      ↓
0000   0000   0010   0011   1100   1111
```

2. Group the binary bits into six-bit groups:

```
0000  0000  0010    0011  1100  1111
└────────┴───────┘  └────────┴───────┘
    ↓         ↓         ↓         ↓
 000000    000010    001111    001111
```

3. Add 32 (binary 100000) to these six-bit binary numerals to form seven-bit ASCII characters:

```
0100000   0100010   0101111   0101111
   ↓         ↓         ↓         ↓
   SP        "         /         /
```

4. 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 $^{S}_{P}$, ", /, and /.

RASTER WRITE = $^{E}_{C}$**RP 6 4**$^{S}_{P}$"//

4526-40

Figure 5-7. Packing Color Index Codes Using Four Bits per Pixel.

RECTANGLE FILL

Sets all the pixels in a rectangle to the specified 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 raster memory space. The range of values for x is 0 to 639; for y, 0 to 511.
Defaults: Omitted = (0,0)

upper-right-corner: xy-coordinate; specifies the opposite corner of that rectangle. The range of values for x is 0 to 639; for y, 0 to 511.
Defaults: Omitted = (0,0)

fill-index: integer; specifies the color index used to fill the rectangle. Must be in the range 0 to 65535.
Defaults: Omitted = 0

The color indices are written into raster memory using the ALU mode and surface specified in the most recent 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 raster memory where the y-value is 480 through 511; however, a Level 0 warning will be generated.

RENAME SEGMENT

Renames (renumbers) an existing segment.

Host Syntax

```
EcSR old-segment-number
     new-segment-number
```

Setup Syntax

```
SGRENAME old-segment-number
         new-segment-number
```

old-segment-number: integer; specifies number of the segment being renamed. Must be in range 1 through 32767.
Defaults: Omitted = Error

new-segment-number: integer; specifies new name (number) for the segment. Must be in range 1 through 32767.
Defaults: Omitted = Error

The existing segment number is changed to the *new-segment-number*. If a segment with the new segment number already exists, an error occurs and the segment is not renamed.

RENEW VIEW

Erases a specified view and redraws all segments visible in that view, including the border and the framing box, if applicable.

Host Syntax

```
EcKN view-number
```

Setup Syntax

```
RENEW view-number
```

view-number: integer; specifies the view to be renewed. Valid values for *view-number* are:

−1	All views
0	The current view
1 — 64	A specific view

Defaults: Omitted = 0

If the view number is in the range 1 through 64, that view (if it exists) is renewed; that is, the viewport for that view is erased, and all the view's visible segments are redrawn. If the specified view does not exist, the terminal detects an error.

Specifying 0 for *view-number* renews the current view. Specifying −1 for *view number* erases the screen and then renews all views in sequence, ending with the current view.

You can group several views together in a view display cluster. However, whenever a RENEW VIEW command is issued for any view in the cluster, all views in the cluster are renewed. For details, see the description of the SET VIEW DISPLAY CLUSTER command.

REPORT DEVICE STATUS

Causes the terminal to send a Device Status Report to the host computer.

Host Syntax

```
EcJQ device-specifier
```

device-specifier: specifies the device whose status is to be reported to the host. Valid devices are:

HC:	Parallel hardcopy port (for 4695 Color Copier)
P0:	RS-232 Peripheral Port 0
P1:	RS-232 Peripheral Port 1

Defaults: Omitted = Error JQ11

This command causes the terminal to send a Device Status Report for the specified device to the host computer. The device may be the color hardcopy interface (HC:), or an RS-232 peripheral port (P0: or P1:).

If a valid but not installed device is specified as the string parameter, the terminal detects Error JQ10 and returns 0 as the status integer in the Device Status Report.

If an invalid *device-code* is specified as the string parameter, the terminal detects error JQ11. Nevertheless, the terminal sends a Device Status Report to the host computer with SpSp as the returned two-character device mnemonic, and 0 as the status integer.

The terminal enters Bypass Mode when sending this report (see ENTER BYPASS MODE).

Device Status Report

This report is sent in response to REPORT DEVICE STATUS. The report has the following format:

signature-character
device-specifier
status-word
EOM-indicator

The following paragraphs detail the parts of the report.

signature-character: the character for non-GIN reports, as set by the most recent SET REPORT SIG CHARS command. If this character is Nu, it is omitted.

device-specifier: two characters; specifying the device whose status is being reported:

$^S{}_P{}^S{}_P$	Invalid device
P0	RS-232 Peripheral Port P0
P1	RS-232 Peripheral Port P1
HC	Color hardcopy interface

status-word: integer; whose binary bits hold status information, see Tables 5-8 and 5-9.

Table 5-8

STATUS INTEGER FOR RS-232 PERIPHERAL PORT

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	B	P

The meanings of these bits (when set to 1 rather than 0) are as follows:

X	Reserved for the future
B	Peripheral port is busy
P	This peripheral port is installed in the terminal. (This does not necessarily mean that there is a peripheral device attached to the terminal port.)

Table 5-9 lists the meaning of bits in the status integer for color hardcopy device.

Table 5-9

STATUS INTEGER FOR COLOR HARDCOPY INTERFACE

B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1	B0
X	X	X	X	X	X	X	X	X	X	X	X	X	X	B	P

The meanings of these bits (when set to 1 rather than 0) are as follows:

X	Reserved for the future
B	The screen pseudo device is busy
P	The screen pseudo device is present

REPORT ERRORS

Causes the terminal to send an Error Report to the host.

Host Syntax

```
EcKQ
```

The REPORT ERRORS command causes the terminal to send an Error Report to the host computer. In that message, the terminal reports the eight most-recently detected error codes, their severity levels, and how many times each error was detected. See the description of the Error Report for details.

The most recent error will be returned first, and the "signature" characters used are those which have been specified for reports.

When the terminal sends a report to the host, Bypass mode is entered. (See ENTER BYPASS MODE Command.)

Error Report

The Error Report is sent in response to REPORT ERRORS. This report is a series of up to eight individual error messages. The Error Report has the following format:

report-for-one-error
terminal-signature-chararacter
EOM-indicator

A single *report-for-one-error* message is sent 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 there are fewer than eight *reports-for-one-error* in the Error Report.

Each *report-for-one-error* is preceded by a *signature-character*, as set by the SET REPORT SIG CHARACTERS command. After the last *report-for-one-error*, the terminal sends a *terminal-signature-character* and an *EOM-indicator*; this signals the end of the Error Report.

The terminal does not send a *signature-character* if it is the NU character. Do not set the *signature-character* or *terminal-signature-character* to NU, since lack of a signature character makes it difficult for the host to parse the Error Report.

Besides the *signature-character*, each *report-for-one-error* includes a four-character error code, a severity level number, and the number of times the terminal has detected that error since power-up or the last REPORT ERRORS command. The error codes and severity levels are described in the Appendix.

The *EOM-indicator* is sent to show between each error message in the report and at the end of the report to show that it is completed. If there are no errors to the report, the terminal sends one *EOM-indicator.*

REPORT GIN POINT

Forces the terminal to send a GIN report to the host.

Host Syntax

```
EcIP  device-function-code
```

device-function-code: integer; enables the specified GIN device and function. See Table 5-6 under the ENABLE GIN command description for valid values. Numeric value –2 reports the current graphics position.
Defaults: Omitted = Error IP11

The REPORT GIN POINT command forces the terminal to return to the host a GIN report for one GIN event, without any operator interaction. The GIN event can be a Locator event, a Pick event, or a Stroke event, depending on the command's *device-function-code* parameter.

If GIN rubberbanding or inking is enabled for that *device-function-code*, then the terminal performs the appropriate rubberbanding or inking function on its display.

The position returned is the location in terminal space of the graphics cursor for the specified *device-function-code.* Specifying –2 for the *device-function-code* causes the terminal to report the current graphics beam position. The format for the report is the same as a Locate Report.

When the terminal sends a report to the host, Bypass mode is entered (see ENTER BYPASS MODE).

Terminal Already Enabled for That Device and Function. If an ENABLE GIN command has already enabled the terminal for the specified *device-function-code*, then the REPORT GIN POINT command serves only to force a GIN event. The terminal behaves as if the operator had initiated the GIN event, except that the ASCII character returned as the "key pressed" part of the GIN report is always the SP character.

Terminal Not Already Enabled for That GIN Device and Function. If the terminal was not already enabled for the specified GIN *device-function-code*, then the following occurs:

1. The terminal executes an implicit ENABLE GIN command for that *device-function-code* and a count parameter of 1.

2. The terminal sends to the host the GIN report sequence for the implicit ENABLE GIN it has just executed. In the report, the xy parameter shows the position in terminal space for the cursor currently assigned to the specified device function combination. In the report, the "key pressed" ASCII-character parameter is the SP character.

The graphics cursor "blinks" momentarily. (The graphics cursor turns on as the terminal executes the implicit ENABLE GIN command. Then it turns off again as the GIN REPORT sequence is sent to the host computer.)

GIN Report Sequence

The GIN report sequence is best represented by a syntax graph as shown in Figure 5-8.

A GIN report sequence is a sequence of reports that the terminal sends the host computer when the terminal has been enabled (by an ENABLE GIN command) for graphics input from a single GIN device.

If more than one GIN device has been enabled, then the corresponding GIN report sequences may be interleaved. That is, the *GIN-report-items* and *final-GIN-report-items* from the various enabled GIN devices may be intermixed. (In that case, "signature characters" are used to distinguish the *GIN-report-items* for one GIN device from the *GIN-report-items* for another GIN device.)

Overall Syntax. The GIN report sequence from a single GIN device consists of a series of *GIN-report-items*, terminated by a *final-GIN-report-item*.

GIN Report Items. The terminal sends a *GIN-report-item* each time a GIN event occurs.

If the GIN device was enabled for the Locator or Pick functions, then a GIN event occurs when the operator presses a keyboard key (for the joydisk GIN device), presses the tablet, pen or a button on the tablet cursor (for the tablet GIN device).

If the GIN device was enabled for the Stroke function, then a GIN event occurs each time a new coordinate is to be sent to the host computer. With the Stroke function, this can happen many times a second.

A typical *GIN-report-item* consists of a signature character, followed by a GIN report. The GIN report format depends on which GIN function was enabled; it is either a GIN Locator Report, GIN Pick Report, or GIN Stroke Report. In addition, an *EOM-indicator* (typically just the ᶜʀ character) may be inserted at the start or end of the *GIN-report-item*.

Signature Characters. The signature characters (*signature-character* and *terminal-signature-character*) are included for convenience when parsing the GIN report sequence. If two or more GIN devices are enabled at the same time, the signature characters serve to distinguish the *GIN-report-items* (and *final-GIN-report-items*) coming from one GIN device from those coming from the other GIN devices.

Also, it is possible, even while GIN is enabled, to issue commands which cause the terminal to send a report to the host computer which is not part of the GIN report sequence. For instance, even while GIN is enabled, a REPORT TERMINAL SETTINGS command can cause the terminal to send a Terminal Settings Report to the host computer. In that case, signature characters may be used to distinguish the Terminal Settings Report from the *GIN-report-items* in the GIN report sequence.

The two signature characters (*signature-character* and *terminal-signature-character*) are determined by the most recent SET REPORT SIG CHARACTERS command for the particular GIN *device-function-code*. For details, see the description of the SET REPORT SIG CHARACTERS command.

GIN Reports. The GIN reports for a single GIN device are all either GIN Locator Reports, GIN Pick Reports, or GIN Stroke Reports. The syntaxes of these are listed in Figure 5-8 as part of the syntax for the GIN report sequence. For more details, however, you should refer to the separate descriptions of the GIN Locator Report, GIN Pick Report, and GIN Stroke Report, later in this command description.

EOM Indicators Within GIN Report Items. The syntax for a *GIN-report-item* includes an optional *EOM-indicator* at the start of that item, and another optional *EOM-indicator* at the end of the item.

By issuing a SET REPORT EOM FREQUENCY: 1 command, you can cause an *EOM-indicator* to be sent at the end of each *GIN-report-item*. That way, each *GIN-report-item* is sent to the host as a single line of text. This may be convenient for writing host routines that parse the GIN report sequence.

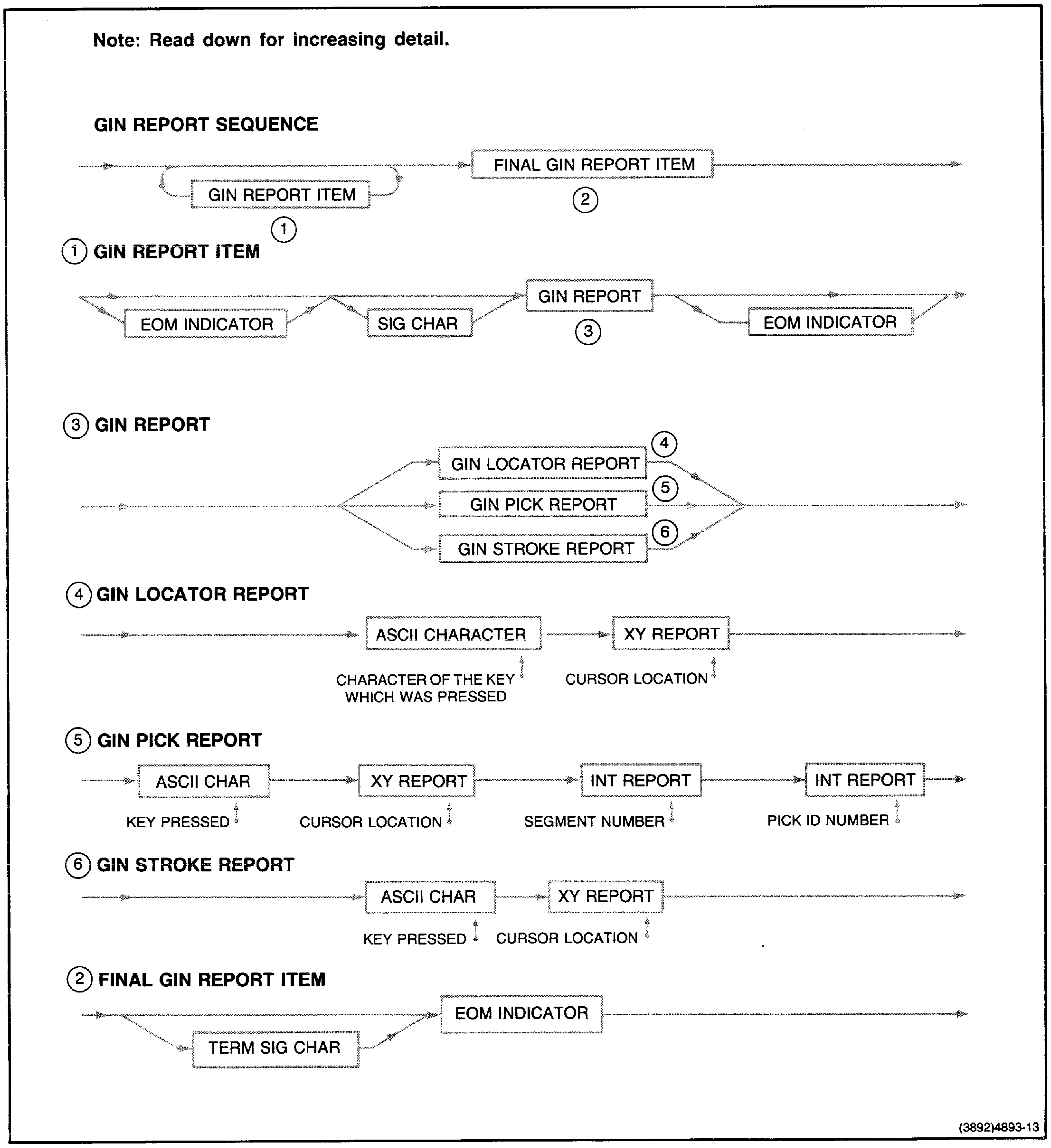


Figure 5-8. Syntax for a Single GIN Report Sequence.

However, you can also have the terminal fit several *GIN-report-items* in the same line of text (or in the same block, if using block mode). To do this, choose the "less frequent" option in the SET REPORT EOM FREQUENCY command. That is, issue a SET REPORT EOM FREQUENCY: 0 command. Also, choose a maximum line length (with the SET REPORT MAX LINE LENGTH command) which is sufficient to hold two or more *GIN-report-items*.

Under these circumstances, the optional *EOM-indicator* at the start of each *GIN-report-item* becomes important. If enough *GIN-report-items* have been sent on the current line, so that even one more *GIN-report-item* would cause the maximum line length to be exceeded, then the terminal sends an *EOM-indicator* at the start of the next *GIN-report-item*. That *EOM-indicator* serves to terminate the current line, so that the maximum line length is not exceeded. The *signature-character* that follows would then be the first character of the next line.

Final GIN Report Item. The terminal sends a *final-GIN-report-item* to the host computer when the graphics input function is disabled. This occurs when the ENABLE GIN command's "count" is exhausted, when the terminal receives a DISABLE GIN command, or when the operator presses the CANCEL key. The *final-GIN-report-item* consists of a *terminal-signature-character*, followed by an *EOM-indicator*.

Terminal Signature Character. The *terminal-signature-character* is a single ASCII character, which serves to notify the host that the GIN report sequence is ended. The *terminal-signature-character*, like the *signature-character* described earlier, is set by the SET REPORT SIG CHARACTERS command.

EOM Indicator in the Final GIN Report Sequence. After the *terminal-signature-character* the terminal always sends an *EOM-indicator*.

GIN Locator Report

The GIN Locator Report has the following format:

key
cursor-location

The following paragraphs detail the parts of the report.

key: ASCII character; the character for the key or tablet switch that the operator pressed to signal a GIN event. If the GIN device is the tablet, then this character is Z, 1, 2, or 3, depending on which button on the four-button cursor the operator presses.

cursor-location: x-y report: the position of the graphics cursor when the operator signalled a GIN event.

When the terminal is enabled for a GIN (graphics input) locator function, the graphics cursor appears and the operator moves the cursor (e.g., with the joydisk) until it is positioned at a location which is to be reported to the host computer. Then the operator signals a GIN locator event (e.g., by pressing a keyboard key). The terminal responds by sending a GIN Locator Report to the host. Each GIN Locator Report is preceded by a signature character.

GIN Pick Report

The GIN Pick Report has the following format:

key
cursor-location
segment-number
pick-ID-number

The following paragraphs detail the parts of the report:

key: ASCII character; the key that the operator pressed to signal the pick event. If the GIN device is the tablet, then this character is Z, 1, 2, or 3, depending on which button on the four-button cursor the operator presses.

cursor-location: xy-report; the position of the graphics cursor at the moment of the pick event.

segment-number: integer-report; the segment number for the segment which the operator picked. The number is 0 if no segment is detected in aperture.

pick-ID-number: integer-report; the Pick identification number for the part of the segment which the operator Picked. The number is 0 if no segment is detected in aperture.

When the terminal is enabled for a GIN (graphics input) Pick function, the graphics cursor appears and the operator moves the cursor (e.g., with the joydisk) until it is positioned at a segment which the operator wishes to "Pick." The operator then signals a Pick event (e.g., by pressing a key).

When the operator signals a pick event, the terminal returns a GIN Pick Report to the host computer. This occurs regardless of whether there actually is a visible, detectable segment within the current pick aperture.

If more than one visible, detectable segment falls within the pick aperture, the segment Picked will be the one with the highest display priority.

For more details, see the descriptions of the SET PICK ID, SET SEGMENT VISIBILITY, SET SEGMENT DETECTABILITY, SET SURFACE VISIBILITY, SET PICK APERTURE, and SET SEGMENT DISPLAY PRIORITY commands.

GIN Stroke Report

The GIN Stroke Report has the following format:

key
cursor-location

The following paragraphs detail the parts of the report.

key: ASCII character; the first point in a stroke. The "key" character is **Z**, **1**, **2**, or **3** depending on which cursor button was pressed. Subsequent points in a stroke are indicated with the letter **J**, and the last point with the letter **O**.

cursor-location: xy-report; the location of the graphics cursor for one point of the stroke.

When an ENABLE GIN command has enabled the graphics tablet for the Stroke function, the GIN reports sent to the host in the GIN report sequence are GIN Stroke Reports.

For each Stroke that the operator performs at the tablet, many GIN Stroke Reports are sent to the host computer.

First Point in the Stroke. The Stroke begins when the operator places the cursor on the tablet surface and presses a button on the cursor. The terminal then sends the first GIN Stroke Report to the host computer.

Subsequent Points. As the operator moves the cursor across the tablet, subsequent GIN Stroke Reports are sent. These report the positions through which the tablet pen or cursor moves. For each of these reports, the *key* field is the ASCII character **J**.

Last Point. The Stroke ends (a) when the ENABLE GIN command's count expires, or (b) when the operator stops pressing the pen against the tablet, removes the cursor from the tablet surface, or releases the button on the cursor. The stroke also ends if the terminal's output buffer is filled up; more about that later.

If the Stroke ends by the ENABLE GIN command's count expiring, then the last point in the Stroke is a valid data point like the ones that preceded it. In that case, the GIN Stroke Report for the last point uses the same *key* as for the preceding points: the letter **J**.

If the Stroke ends by an operator action (such as removing the pen from presence), it is possible that the last xy-report does not represent a valid coordinate. In that case, the final GIN Stroke Report includes a *key* field which is the ASCII letter **O**. This notifies the host program that it should not rely on the accuracy of the associated xy-report.

Filling Up the Output Buffer. When using the Stroke graphics input function, it is easy to digitize points faster than the terminal can send the corresponding GIN Stroke Reports to the host computer. When this happens, the terminal's output buffer can overflow. (You may be able to avoid this condition by using a high baud rate, or by using stroke filtering in order to digitize points less frequently.)

If the terminal's output buffer is full, then the terminal can accept no more graphics input data until some of the characters in that buffer have been sent to the host computer. With the buffer full, attempting to enter more points (a) causes the current stroke to end, and (b) causes the terminal to sound its bell. The bell serves to warn the operator to pause before digitizing more points. (The pause gives time for the terminal to transmit characters to the host, thereby freeing memory in the output buffer.)

REPORT PORT STATUS

Sends a Port Status Report to the host or screen.

Host Syntax

```
EcPQ port
```

port: integer; specifies the peripheral port whose status is to be reported. Valid devices are:

P0:	RS-232 Peripherial Port P0
P1:	RS-232 Peripherial Port P1

Defaults: Omitted = Error PQ11

When the REPORT PORT STATUS command is received as an escape sequence, the terminal sends a Port Status Report for the specified RS-232 peripheral port to the host computer.

When the terminal sends a report to the host, Bypass mode is entered. (See ENTER BYPASS MODE.)

If the *port* parameter is invalid (neither P0:, nor P1:), then the terminal detects a type PQ11 error. Nevertheless, it still sends a Port Status Report to the host computer. That report, however, is abbreviated; its port ID code consists of two SP characters, and the port information is omitted. For details, see the following description of the Port Status Report.

Port Status Report

The Port Status Report has the following format:

EOM-indicator
signature-character
port-ID-code
port-information
EOM-indicator

EOM-Indicator. An *EOM-indicator* may be sent at the start of the Port Status Report. This optional *EOM-indicator* is provided because of the terminal's "maximum report line length" feature. This *EOM-indicator* is included in the Port Status Report only if it is needed to prevent the current maximum line length from being exceeded. (See the description of the SET REPORT MAX LINE LENGTH command for details.)

An *EOM-indicator* is included at the beginning of a Port Status Report only if both the following conditions are met:

- At least one character has already been sent on the current line (that is, since the last *EOM-indicator*).
- If the remainder of the *port-status-report* could cause the current maximum line length to be exceeded.

Signature-Character. The *signature-character* is provided for easier parsing of the report by the host computer. (If graphics input is active, the signature character allows the host to distinguish this report from GIN reports which are part of a GIN report sequence.) The signature character in the Port Status Report is the current *signature-character* for non-GIN reports, as set by the most recent SET REPORT SIG CHARS command. If the current *signature-character* is NU, then it is omitted from the Port Status Report.

Port-ID-Code. Next comes a two-letter code: **P0**, **P1**, or SPSP. This names the RS-232 peripheral port to which the Port Status Report pertains. These two characters are sent to the host as *character-reports.*

A *port-ID-code* of SPSP indicates that the REPORT PORT STATUS command had an invalid *port-specifier* string. If that is the case, then the following *port-information* parameter is omitted.

Port Information. The *port-information* consists of a series of *int-reports*, *string-reports*, and *int-array-reports.* These contain the current values of the peripheral port's parameters: baud rate, parity, number of stop bits, etc.

The *device-driver-name string-report* that is part of the *port-information* is always ten characters long.

Final EOM-Indicator. The Port Status Report ends with an *EOM-indicator*. This final *EOM-indicator* is always sent; it helps ensure that the host applications program actually receives the Port Status Report in a timely manner. (In some host operating systems, the user application program does not receive a message from the terminal until the terminal sends a CR or other end-of-message indicator.)

REPORT SEGMENT STATUS

Causes the terminal to send a Segment Status Report to the host.

Host Syntax

```
EcSQ segment-number,
     status-codes
```

segment-number: integer; specifies the segment number in the range (–3 to 32767) for which you want information.

–3	All segments that match the current matching class
–2	The default values for segments not yet defined
–1	All segments in the range 1 through 32767
0	The crosshair cursor
1 to 32767	A specific segment

Defaults: Omitted = 0

status-codes: character array; specifies the kind of information to be returned for the specified segment. Valid array characters are:

A	Segment classes
D	Detectability
H	Highlighting mode
I	Image transform parameters
M	Writing mode
P	Pivot point
S	Display priority number
V	Visibility
X	Position

Defaults: Omitted = Empty array

The REPORT SEGMENT STATUS command causes the terminal to send a Segment Status Report to the host computer.

When the terminal sends a report to the host, it enters Bypass mode. (See ENTER BYPASS MODE.)

This command is intended to be issued by the host computer. It should not be typed in by the operator with the terminal in Local mode.

Segment Status Report

The Segment Status Report has the following format:

report-for-one-segment
terminal-signature-character
EOM-indicator

The following paragraphs detail the parts of the report.

Report-for-One-Segment. Each *report-for-one-segment* describes the attributes for one segment. It may begin with an optional EOM-indicator. The EOM-indicator is provided because of the terminal's "maximum line length feature". If too many characters have been sent on the current line, so that sending the remainder of the report-for-one-segment would cause the maximum line length to be exceeded, then the terminal ends the current line with an EOM indicator. The signature-character that follows would then be the first character of the next line of text.

This EOM indicator is sent only if the following conditions are met:

- At least one character has already been sent on this line. (That is, at least one character has already been sent since the last EOM indicator.)
- If this *report-for-one-segment* without the EOM indicator would cause the maximum line length to be exceeded.
- The most recent SET REPORT EOM FREQUENCY command specified "more frequently" rather than "less frequently".

Terminal-Signature-Character. The *terminal-signature-character* is determined by the most recent SET REPORT SIG CHARACTERS command for non-GIN reports. (See the description of the SET REPORT SIG CHARACTERS command for details.) The *terminal-signature-character* is provided as a convenience for the host routine which parses the Segment Status Report; it marks the end of the report. If the *terminal-signature-character* is NU, it is omitted from the segment-status-report.

Final EOM-Indicator. After sending the *terminal-signature-character,* the terminal ends the Segment Status Report with an EOM indicator. This EOM indicator is always sent; it helps to ensure that the host applications program receives the preceding characters in a timely manner. (In most host operating systems, the user program does not actually receive a message from the terminal until the message ends with a CR.)

The signature character is provided as a convenience for the host program's parsing routine. It serves to mark the beginning of each *report-for-one-segment*. The signature character is a single ASCII character, determined by the most recent SET REPORT SIG CHARACTERS command for non-GIN reports. (See the SET REPORTS SIG CHARACTERS command for details.) If the signature character is NU, it is omitted.

After the signature character comes an integer report: the segment number (or error code) for the particular segment whose attributes are being reported. If the character-array parameter in the REPORT SEGMENT STATUS command was empty, then the segment number is the only item reported in the *report-for-one-segment*. An error code (described later) is subsituted for the segment number if an invalid segment number or segment attribute codes, or if the segment specified does not exist.

Following the segment number, each *report-for-one-segment* contains zero or more Segment Attribute Reports. There is one Segment Attribute Report for each letter in the character-array parameter of the REPORT SEGMENT STATUS command.

Each Segment Attribute Report contains information about one of the segment's attributes, and begins with the code letter for that attribute. For instance, a Segment Classes Report begins with the letter A, a Visibility Report begins with the letter V, and a Position Report begins with the letter X.

REPORT SYNTAX MODE

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

Host Syntax

```
Ec#!0
```

This command has the same effect as a REPORT TERMINAL SETTINGS command issued for the SELECT CODE command (as if **EcIQ%!** was sent from the host). See REPORT TERMINAL SETTINGS.

This command is recognized in Ansi, Tek, and VT52 modes.

REPORT TERMINAL SETTINGS

Causes the terminal to send a Terminal Settings Report to the host computer.

Host Syntax

```
EcIQ inquiry-code
```

inquiry-code: integer; a two-character parameter containing the two-letter opcode for an escape-sequence command or a special two-character inquiry code for other information about the terminal.

This general-purpose inquiry command tells the terminal to send a Terminal Settings Report to the host computer.

When the terminal sends a report to the host, it enters Bypass mode. (See the description of the ENTER BYPASS MODE command.)

Besides the opcodes for commands, special inquiry codes can be given in the REPORT TERMINAL SETTINGS command. These codes are listed in Table 5-10.

Table 5-10

SPECIAL INQUIRY CODES

Code	Associated Parameter Reports
?M	integer-report: available-memory integer-report: largest-contiguous-block (The available memory, and the size of the largest contiguous block, are reported as a number of 16-byte units of memory.)
?T	integer-report: model-number-code
00	integer-report: standard-firmware-version-number

Terminal Settings Report

The Terminal Settings Report has the following format:

EOM-indicator
signature-character
opcode-report
parameter-report
EOM-indicator

EOM-Indicator. An optional *EOM-indicator* is provided at the start of the terminal-settings-report because of the terminal's "maximum line length" feature. (See the description of the SET REPORT MAX LINE LENGTH command for details.) This *EOM-indicator* is only sent if not sending it would cause the terminal's maximum line length to be exceeded.

Signature-Character. The *signature-character* (signature character) is sent only if it is not $^{N}_{U}$. This character is the current *signature-character* for non-GIN reports, as set by the most recent SET REPORT SIG CHARACTERS command.

Opcode-Report. Next comes the *opcode-report*, consisting of two *character-reports.* The two characters being reported are the same two characters which were used in the REPORT TERMINAL SETTINGS command; they comprise either an opcode for one of the terminal's commands, or else a special inquiry code.

However, if the REPORT TERMINAL SETTINGS command specifies an op code for a command which does not exist, or which is not installed in the teminal, then the *opcode-report* is $^{S}_{P}{}^{S}_{P}$.

The special inquiry codes are listed in Table 5-10. For each inquiry code, the table also shows the *parameter-reports* which are included in that inquiry code's Terminal Settings Report.

Parameter-Report. The command specified by opcode in the *opcode-report* has in its syntax a number of parameters. The current values of these parameters are returned, in order, in the parameter-reports.

Each *parameter-report* is of an appropriate type for the parameter being reported. For instance, consider a Terminal Settings Report for the terminal's baud rate. The SET BAUD RATE command (op code NR) has two parameters of type *integer*. The terminal sends the values of *integer* parameters to the host using the *integer-report* syntax. Therefore, in the corresponding Terminal Settings Report, there are two *integer-reports.*

For special inquiry codes, use the *Parameter Reports* listed earlier in Table 5-10.

Examples

Reporting Baud Rates. The REPORT TERMINAL SETTINGS: **NR** command requests the terminal to send a report of its current baud rate settings. (This is because it is the SET BAUD RATES command which has the opcode NR.)

Assume that the terminal is not in Block mode, that its current *EOL-string* is the single character, $^{C}_{R}$, and that the *signature-character* for non-GIN reports is *A*. Assume also that the terminal is set to transmit and receive at 1200 baud. In that case, the report which the terminal sends to the host is:

	EOM-indicator (usually omitted)
	signature-character
	opcode-report
parameter-report:	*first-baud-rate*
parameter-report:	*second-baud-rate*
	EOM-indicator
	= **A**
	NR
integer-report:	1200
integer-report:	1200
	$^{C}_{R}$
	= **ANR! + 0! + 0**$^{C}_{R}$

Reporting the Amount of Available Memory. To request a report on the amount available memory, the host sends a REPORT TERMINAL SETTINGS: **?M** command. If the *signature-character* and *EOL-string* are as in the previous example, and if the terminal is not in Block mode, then the report which the terminal sends the host is as follows:

	EOM-indicator (usually omitted)
	signature-character
	opcode-report
parameter-report:	*available-memory*
parameter-report:	*largest-block*
	EOM-indicator
	= **A**
	?M
integer-report:	*available-memory*
integer-report:	*largest-block*
	$^{C}_{R}$

Exceptions

For a few commands (or opcodes), the meanings of the parameters reported in the Terminal Settings Report differ from the meanings of the parameters when sending those commands to the terminal. These commands are:

PROMPT MODE (opcode NM)
LOCK VIEWING KEYS (opcode RJ)
SET SURFACE COLOR MAP (opcode TG)
SET VIEW DISPLAY CLUSTER (opcode RQ)

PROMPT MODE Command. Prompt mode can be turned on with a parameter of 1 or 2. However, the terminal only reports whether the Prompt mode is on (1) or off (0).

LOCK VIEWING KEYS Command. This command takes the form ^{E}c**RJ** *integer*. Therefore, a REPORT TERMINAL SETTINGS: **RJ** command causes the terminal to send a Terminal Settings Report which has one *parameter-report*, of the *integer-report* type:

```
                  EOM-indicator
                  signature-character
                  RJ
integer-report:   viewing-key-status
                  EOM-indicator.
```

In this report, however, the *viewing-key-status* integer can assume more values than just zero and one. It can assume values from zero to three. The meanings of these are as follows:

0	The terminal is in not in Framing mode. (That is, it is neither in Zoom mode nor in Pan mode.) Moreover, the viewing keys are not locked. (This does not preclude the entire keyboard's being locked as a result of a lock-keyboard command.)
1	The terminal is in Zoom mode.
2	The terminal is in Pan mode.
3	The viewing keys are locked. (Therefore, the terminal is neither in Zoom mode nor in Pan mode.)

SET SURFACE COLOR MAP Command. This command has the syntax ^{E}c**TG** *integer-array*. Therefore, a REPORT TERMINAL SETTINGS: **TG** command causes the terminal to send to the host a Terminal Settings Report with the following syntax:

```
                        EOM-indicator
                        signature-character
                        TG
      integer-report:   number-of-surfaces
integer-array-report:   color-info
                        EOM-indicator
```

Here, the *integer-array* tells the number of surfaces currently defined, while the *integer-array-report* contains information about the background color mixture, and about the color mixtures for each of the color indices on each of the surfaces. (This is different from the meaning these parameters have when they are in SET SURFACE COLOR MAP command sent from the host to the terminal.)

```
integer-array-report:color-info
  = integer-report:number-of-integer-reports-to-follow
      triple-report:background-color
                    colors-for-one-surface...

colors-for-one-surface
  = integer-report:  negative-surface-number
      triple report: color-coordinates-for-one-color-index...

                   {HLS-triple-report}
triple-report  =   {RGB-triple-report}
                   {CMY-triple-report}

HLS-triple-report  =  integer-report: hue-angle-in-degrees
                      integer-report: lightness-percentage
                      integer-report: saturation-percentage

RGB-triple-report  =  integer-report: red-percentage
                      integer-report: green-percentage
                      integer-report: blue-percentage

CMY-triple-report  =  integer-report: cyan-percentage
                      integer-report: magenta-percentage
                      integer-report: blue-percentage
```

The HLS, RGB, or CMY color coordinate system is used in the report, depending on which color specifying mode was selected by the most recent SET COLOR MODE command.

Suppose, for instance, that HLS color coordinates are being used. (This is the default when the terminal is turned on.) Consider the following *integer-array-report*:

integer-array-report:*color-info*	=	integer-report:	17
		integer-report:	0
		integer-report:	0
		integer-report:	0
		integer-report:	−1
		integer-report:	120
		integer-report:	50
		integer-report:	100
		integer-report:	−2
		integer-report:	180
		integer-report:	50
		integer-report:	100
		integer-report:	240
		integer-report:	50
		integer-report:	100
		integer-report:	0
		integer-report:	100
		integer-report:	0

Here, the first *integer-report* says that there are 17 *integer-reports* to follow in the *integer-array-report*.

The next three *integer-report*s say that the background color is black (hue zero degrees, lightness zero, saturation zero).

The following four *integer-report*s carry the numbers −1, 120, 50, 100. Thus, on surface one, color index one is displayed as a red color (hue 120 degrees, lightness 50, saturation 100). Since only one set of three coordinates follows the −1, there must be only one non-zero color index for that surface. In other words, surface one has only one bit plane assigned to it.

Likewise, the next ten *integer-report*s carry the numbers -2, 180, 50, 100, 240, 50, 100, 0, 100, 0. These give three sets of color coordinates for surface number two. On that surface, color index one is displayed as yellow (hue 180, lightness 50, saturation 100), color index two as green (hue 240, lightness 50, saturation 100), and color index three as white (hue 0, lightness 100, saturation 0). Since three sets of color coordinates are reported for surface two, that surface must have two bit planes assigned to it.

SET VIEW DISPLAY CLUSTER Command. This command has the syntax ${}^{E}_{C}$**RQ** *integer-array*. Therefore, the command ${}^{E}_{C}$**IQRQ** causes the terminal to send to the host a Terminal Settings Report with the following syntax:

EOM-indicator
signature-character
RQ
integer-array-report
EOM-indicator

Here, the *integer-array-report* tells how views are grouped into view display clusters. For the purposes of this report, the clusters are assigned numbers. The first number in the *int-array-report* tells to which cluster, if any, view number one is assigned. Likewise, the second number in the array tells to which cluster view two is assigned, — and so on. If a view has not been assigned to any display cluster, then "cluster number zero" is reported for that view.

Suppose, for instance, that the signature character for non-GIN reports is the tilde (~), and that the host has issued the character sequence, ${}^{E}_{C}$**IQRQ**. (This is a REPORT TERMINAL SETTINGS command which inquires about the view display cluster settings.) One possible response from the terminal would be:

~ **RQ**${}^{S}_{P}{}^{S}_{P}$**6**${}^{S}_{P}{}^{S}_{P}$**1**${}^{S}_{P}{}^{S}_{P}$**1**${}^{S}_{P}{}^{S}_{P}$**1**${}^{S}_{P}{}^{S}_{P}$**0**${}^{S}_{P}{}^{S}_{P}$**2**${}^{S}_{P}{}^{S}_{P}$**2**${}^{C}_{R}$

Here, the tilde (~) is the *signature-character*, the characters **RQ** signify that the report is for the RQ opcode, and the final ${}^{C}_{R}$ is the *EOM-indicator*. The other characters comprise an *int-array-report*, as follows:

${}^{S}_{P}{}^{S}_{P}$**6**	=	int-report: 6	(The array has six items)
${}^{S}_{P}{}^{S}_{P}$**1**	=	int-report: 1	(View 1 is in Display Cluster 1)
${}^{S}_{P}{}^{S}_{P}$**1**	=	int-report: 1	(View 2 is in Display Cluster 1)
${}^{S}_{P}{}^{S}_{P}$**1**	=	int-report: 1	(View 3 is in Display Cluster 1)
${}^{S}_{P}{}^{S}_{P}$**0**	=	int-report: 0	(View 4 is not in any display cluster)
${}^{S}_{P}{}^{S}_{P}$**2**	=	int-report: 2	(View 5 is in Display Cluster 2)
${}^{S}_{P}{}^{S}_{P}$**2**	=	int-report: 2	(View 6 is in Display Cluster 2)

Thus, the Terminal Settings Report tells the host these things:

- The highest-numbered view which is in a view display cluster is view number six.
- Views one, two, and three are in the same display cluster.
- Views five and six are in the same display cluster.
- View four is not in any view display cluster.

REPORT 4010 STATUS

Causes the terminal to emulate a 4010-style terminal by sending a 4010 Status Report.

Host Syntax

```
EcEQ
```

This command also terminates 4010 GIN mode and returns the terminal to Alpha mode.

4010 Status Report

This report is sent in response to REPORT 4010 STATUS. The report has two forms, depending on whether the terminal is in GIN mode when the command is sent.

If the terminal is *not* in GIN mode, the report has the following format:

terminal-status
alpha-cursor-position
EOM-indicator

If the terminal is in GIN mode, the report has the following format:

graphics-cursor-position
EOM-indicator

The following paragraphs detail the parts of the report.

Terminal-Status. The status of the terminal is encoded into the seven bits of an ASCII character, as follows:

b7	b6	b5	b4	b3	b2	b1
0	1	HCU	NOLI	GRAPH	0	1

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 hardcopy unit is attached to the terminal and is ready to accept a copy request; otherwise this bit is set to 1.

Bits 4 and 5 indicate the No Linear Interpolation (NOLI) and Graph mode status as follows:

NOLI	GRAPH
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 does not occur

For example, if the terminal (1) has a hardcopy unit attached, (2) is ready for a hardcopy command, and (3) is in Vector mode, the bits sent are:

```
0 1 0 1 0 0 1
```

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. The terminal reports both cursor positions in the following format:

Hi-X
Lo-X
Hi-Y
Lo-Y

The Hi-X, Lo-X, Hi-Y,and Lo-Y characters follow the same general coding method described under the heading *XY-Coordinates in Host Syntax* earlier in this section, with two important differences:

- The first two bits of each character are set to 01.
- Only the ten most significant bits are included in the report.

Figure 5-9 illustrates the binary process of decoding the ASCII characters that report that the cursor is at (1000,2000). To report these coordinates the terminal sends the following characters:

```
':/4
```

The host program must convert each character to equivalent numbers (ASCII Decimal Equivalents). Figure 5-9 shows the binary representation of these numbers. The first two bits are always 01, confirming that these are report characters. To get rid of the 01 prefix, subtract decimal 32 from each number (Hi-X = 39 – 32 = 7). The remaining values combine to form the two decimal xy-coordinates:

- Multiply the Hi-X value by decimal 128 and save the product: (7 * 128 = 896).
- Multiply the Lo-X value by decimal 4 and save the product: (26 * 4 = 104).
- Add the products, yielding the x-coordinate: (896 + 104 = 1000).
- Repeat the previous three steps with the y-values to find the y-coordinate.

Notice that the last two bits of the cursor position are not reported and can be assumed to be (00).

EOM-Indicator. The EOM (End Of Message) character may be sent to show that the 4010 Status Report message is complete. However, it is not an essential part of the message.

RESET

Returns the terminal to its power-up condition.

Host Syntax

```
EcKV
```

Setup Syntax

```
RESET
```

The RESET command initializes the terminal to its power-up condition. It is equivalent to pressing the RESET button, or to turning the terminal off and then turning it on again.

NOTE

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.

Therefore, if you issue a RESET command from the host, you should wait at least 15 seconds before sending other commands or data to the terminal.

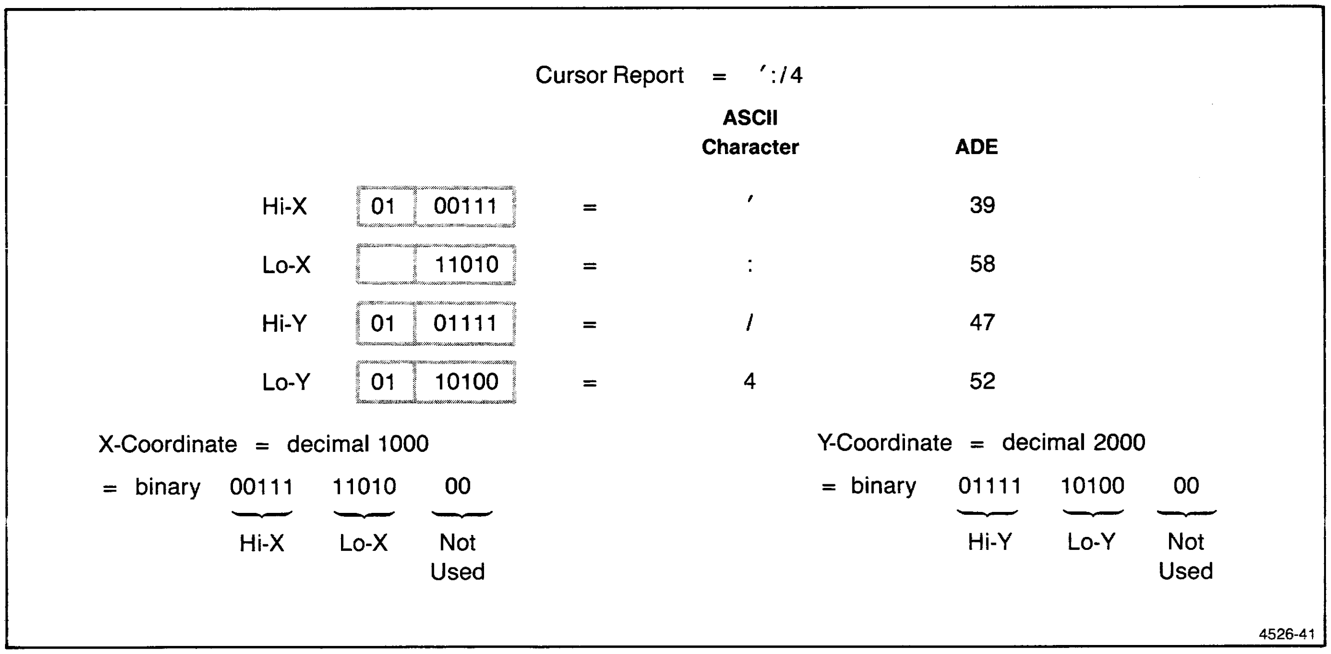


Figure 5-9. The Cursor Position Report.

RUNLENGTH WRITE

Loads color indices into the pixel viewport.

Host Syntax

```
EcRL runcode-array
```

Setup Syntax

```
PXRUNLENGTHWRITE runcode-array
```

runcode-array: integer-array; uses runcodes to write color indices to the pixel viewport. Each runcode can range from 0 to 65535.
Defaults: Omitted = Empty Array

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 array. The color indices are written into raster memory using the ALU mode and surface specified in the most recent BEGIN PIXEL OPERATIONS command. As each pixel is assigned a color index, the pixel beam position moves so that it points at the next pixel to the right on the same line. On encountering the right edge of the pixel viewport, the pixel beam position wraps around to point to the pixel at the left edge of the pixel viewport on the line below; however, if the pixel beam position is on the bottom line, the beam position wraps around to the left edge of the top line of the pixel viewport. 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.

Each runcode includes two numbers packed together: one is a color index, and the other is the number of pixels which are to be set to that color index. The runcodes are packed using the form

Runcode = number-of-pixels $*$ 2^n + color-index

where n = number-of-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 6; then the value of n is 4.

If the number of bits per pixel is greater than the number of bit planes on the current surface, the surface receives only the most significant bits of each color index sent. For instance, if the number of bits per pixel is 3 and the current surface has only two bit planes, a color index with a binary value of 110 is written to the current surface as 11. In the case of six bits per pixel, if color index 8 (binary 001000) is sent to a surface that has three bit planes, the bits 1000 are sent and written to the surface as 100.

SAVE NONVOLATILE PARAMETERS

Saves nonvolatile parameters that have been altered.

Host Syntax

```
EcKU
```

Setup Syntax

```
NVSAVE
```

This command causes the terminal to write to its nonvolatile memory all parameters that have been changed since the last SAVE NONVOLATILE PARAMETERS command was issued. The values of those parameters are restored at power-up and are referred to in this manual as the *Power-Up Defaults.*

Nonvolatile memory means memory that is retained after the terminal is turned off. Parameters that are saved in nonvolatile memory are referred to in this manual as *saved in memory.*

This command saves only parameters that have changed since the last time this command was issued.

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) that states that there is a "nonvolatile hardware error." When that occurs, parameters are reset to factory default the next time the terminal is powered up.

SELECT CODE

Causes the terminal to recognize Ansi, Tek, or VT52 mode command syntax. Also used to select Edit mode.

Host Syntax

```
EcMP%! syntax
```

Setup Syntax

```
CODE  syntax
```

syntax: integer; (keyword in Setup syntax); selects the following modes for Host or Setup.

Host	Setup	
0	TEK	Selects Tek mode syntax
1	ANSI	Selects Ansi mode syntax
2	EDIT	Selects Ansi mode syntax for Edit mode
3	VT52	Selects VT52 mode syntax

Defaults:	Factory	= 0 (Tek mode)
	Power-Up	= Saved in memory
	Omitted	= 0 (Tek mode)

The syntax of Tek, Ansi, and VT52 mode commands are not compatible. If you are using 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.

In Ansi mode, the terminal:

- Recognizes Ansi mode commands from the host

In Tek mode, the terminal:

- Recognizes 4100-style commands from the host

In Edit mode, the terminal:

- Recognizes Ansi mode commands from the host
- Sets Origin mode to Absolute
- Sets dialog area and dialog 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

SELECT FILL PATTERN

Specifies the fill pattern for subsequent panels.

Host Syntax

```
EcMP  fill-pattern-number
```

Setup Syntax

```
FILLPATTERN  fill-pattern-number
```

fill-pattern-number: integer; must be in the range −15 through 174:

−15 — 0	Fill a panel with a solid color indicated by the negative value of a color index (for example, −3 means fill with color index 3's color)
1 — 16	Specify predefined patterns
17 — 49	Are invalid and generate an error
50 — 174	Specify predefined dither patterns

Defaults:	Factory	= −1
	Power-Up	= −1
	Omitted	= 0

For example, here's how you would select Fill Pattern 16 (encoded as **A0**) in Host Syntax:

EcMPA0

Refer to the appendix in the Operators Manual that contains examples of the fill patterns associated with the fill pattern numbers.

SELECT HARDCOPY INTERFACE

Selects the copier type to be used in the HARDCOPY command.

Host Syntax

```
EcQD copier-type
```

Setup Syntax

```
HCINTERFACE copier-type
```

copier-type: integer; use one of the following:

0	Selects a monochrome printer
1 or 2	Selects the TEKTRONIX 4695 Color Copier

Defaults: Factory = 2
Power-Up = Saved in memory
Omitted = 0

The monochrome selection is for compatibility with black-and-white printers that use a parallel interface. Selecting a monochrome copy type allows only regular-sized dialog copies.

SELECT VIEW

Specifies which view will be the current view.

Host Syntax

```
EcRC view-number
```

Setup Syntax

```
VSELECT view-number
```

view-number: integer; specifies the view to be selected. Valid values for *view-number* are:

–1	The next lower-numbered view
0	The next higher-numbered view
1 — 64	A specific view

Defaults: Factory = 1
Power-Up = 1
Omitted = 0

A view is defined by a set of viewing parameters and (possibly) a set of visible segments. There may be up to 64 views defined and each one may be independent of the others. Only one view can be selected to be current at a time. All actions, such as unretained graphics, segments made visible, renew commands, etc., go into this current view. When a new view is designated as current, the attributes of the previous view are remembered until the next time that view is selected.

If the view has never been selected before, a new view (with the specified view number) is created.

Creating a New View. When a new view is created in this manner, the view's parameters are set as follows:

Window:	The window for the new view is the same as that for the previous view; that is, it has the same lower-left and upper-right corners in terminal space.
Viewport:	The viewport for the new view is on the same surface, and has the same lower-left and upper right corners, as the viewport for the previous view.

Surface:	The surface on which the viewport is displayed is the same as that for the previous view. (It can be changed with the SET VIEW ATTRIBUTES command.)
Segments:	All segments that already exist remain in existence. However, none of these segments is visible in the new view. (Segment visibility applies only to a particular view; see the SET SEGMENT VISIBILITY command.)
Indices:	The new view's erase index and border index are the same as those for the previous view. (These indices, and the surface number, can be changed with a SET VIEW ATTRIBUTES command.)
Border:	The visibility of the new view's border is the same as for the previous view. (It can be changed with a SET BORDER VISIBILITY command or the BORDER key.)

Other View Numbers. Specifying View 0 causes the terminal to select the next higher-numbered view of the existing numbered views. If there is no higher-numbered view in existence, then the lowest-numbered view is selected. This method of selecting the next view is equivalent to pressing the NEXTVIEW key, except that the view's border does not blink.

Specifying View –1 selects the next lower-numbered view of the existing numbered views. If no such lower-numbered view exists, then the highest-numbered view is selected. This method is equivalent to pressing CTRL NEXTVIEW, except that the view's border does not blink.

Default View. The default view, which is created at power-up (or when a DELETE VIEW command with –1 is issued), has these attributes:

View number:	1
Window:	x = 0 to x = 4095, y = 0 to y = 3136
Viewport:	x = 4095, y = 3071
Surface number:	1
Border:	invisible
Graphics beam position:	(0,3071)
Wipe Index:	0

SET ALPHA CURSOR INDICES

Assigns specified color indices to the alpha cursor.

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. The values 0 through 7 correspond to a color index; a value greater than 7 sets *first-index* to 7.

Defaults:		
	Factory	= 1
	Power-Up	= Saved in memory
	Omitted	= 0

second-index: integer; specifies the second color for the alpha cursor; must be in the range 0 to 65535. The values 0 to 7 correspond to a color index. Values greater than 7 set *second-index* to 7.

Defaults:		
	Factory	= 0
	Power-Up	= Saved in memory
	Omitted	= 0

The alpha cursor appears on the screen where the next alphanumeric character will be displayed. If the *second-index* is is a different color than for the *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.

SET ALPHATEXT FONT

Selects font to be used for alphatext.

Host Syntax

```
Ec font-code
```

font-code: small integer; the SI character selects the G0 character set, and SO selects the G1 character set. Regardless of the alphtext font selected, in Setup mode, the terminal uses the ASCII font.
Defaults: Power-Up = G0 character set

SET BACKGROUND COLOR

Sets the color of the background surface which is behind all of the transparent writing surfaces.

Host Syntax

```
EcTB first-color-coordinate,
     second-color-coordinate,
     third-color-coordinate
```

Setup Syntax

```
CBACKGROUND first-color-coordinate,
            second-color-coordinate,
            third-color-coordinate
```

first-color-coordinate: integer; specifies a color coordinate (H, R, or C) depending on the *color-specifying-mode* in the most recent SET COLOR MODE command. The valid range of parameter values for each color coordinate system is:

HLS	-32768 to 32767
RGB	0 to 100
CMY	0 to 100

Defaults: Factory = 0
Power-Up = 0
Omitted = 0

second-color-coordinate integer; specifies a color coordinate (L, G, or M) depending on the *color-specifying-mode* in the most recent SET COLOR MODE command. The valid range of parameter values for each color coordinate system is:

HLS	0 to 100
RGB	0 to 100
CMY	0 to 100

Defaults: Factory = 0
Power-Up = 0
Omitted = 0

third-color-coordinate integer; specifies a color coordinate (S, B, or Y) depending on the *color-specifying-mode* in the most recent SET COLOR MODE command. The valid range of parameter values for each color coordinate system is:

HLS	0 to 100, or 1000 to 1100
RGB	0 to 100, or 1000 to 1100
CMY	0 to 100, or 1000 to 1100

Defaults: Factory = 0
Power-Up = 0
Omitted = 0

The three color coordinates are either HLS, RGB, or CMY according to the *color-specifying-mode* in the most recent SET COLOR MODE command. The SET SURFACE COLOR MAP command can also be used to set the background color.

You can specify a blinking color by adding 1000 to the value of the *third-color-coordinate* parameter. The color blinks by alternating between black and the specified color at a rate of (0.75 sec on, 0.75 sec off). For example, in HLS mode a normal red background is indicated by (120, 50, 100), and a blinking red background is given by (120, 50, 1100).

NOTE

If you specify a SUBTRACTIVE overlay mode in the SET COLOR MODE command, then you should also specify a background color of white (or some other light color) with the SET BACKGROUND COLOR command.

SET BACKGROUND INDICES

Specifies the color indices for the character backgrounds of string precision graphtext (and alphatext outside the dialog area); also specifies the color index used for the gaps in dashed lines.

Host Syntax

```
EcMB  text-background-index
      dash-gap-index
```

Setup Syntax

```
BACKINDICES  text-background-index
             dash-gap-index
```

text-background-index: integer; specifies the background index for string precision graphtext and alphatext which is not displayed in the dialog area. Must be in the range -2 through 32767.

-2	Specifies the wipe index for the current viewport
-1	Leaves the character background pixels unchanged
0 — 32767	Specifies a particular color index

Defaults: Factory = -1
Power-Up = Agrees with mode saved in memory for SET GRAPHICS AREA WRITING MODE. (Overstrike mode equals -2, Replace mode equals -1.)
Omitted = 0

dash-gap-index: integer; determines the color index for the gaps in dashed lines. Must be in the range -2 to 32767.

-2	Specifies the wipe index for the current viewport
-1	Leaves the line-gap pixels unchanged
0 — 32767	Specifies a particular color index

Defaults: Factory = -1
Power-Up = -1
Omitted = 0

Specifying -2 for the *text-background-index* is like specifying Replace mode in the SET GRAPHICS AREA WRITING MODE command. Specifying -1 for the *text-background-index* is like specifying Overstrike mode in the SET GRAPHICS AREA WRITING MODE command.

NOTE

The SET GRAPHICS AREA WRITING MODE and SET BACKGROUND INDICES commands both affect how alphatext is displayed in the graphics area. Thus, each of these commands supersedes the effect of the other.

SET BAUD RATES

Sets the terminal's transmit and receive baud rates.

Host Syntax

```
EcNR transmit-data-rate
     receive-data-rate
```

Setup Syntax

```
BAUDRATE transmit-data-rate
         receive-data-rate
```

transmit-data-rate: integer; 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, 2400, 4800, 9600, 19200, and 38400.

Defaults:	Factory	= 2400
	Power-Up	= Saved in memory
	Omitted	= Error

receive-data-rate: integer; 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
	Power-Up	= Saved in memory
	Omitted	= Same as *transmit-baud-rate*

The transmit and receive parameters must be the same if you set the baud rate to 38400; otherwise the transmit and receive rates do not need to be the same.

SET BORDER VISIBILITY

Controls the visibility of a border drawn around the current view's viewport.

Host Syntax

```
EcRE border-visibility-mode
```

Setup Syntax

```
BORDER border-visibility-mode
```

border-visibility-mode: integer; specifies whether the border of the current view is visible or invisible. Must be chosen from the following:

0	Invisible
1	Visible
2	Toggles the border visibility (if visible, it becomes invisible; if invisible, it becomes visible)

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

The border is drawn as a solid line, just within the viewport. It is drawn in the color index specified by the SET VIEW ATTRIBUTES command for that view. (If no SET VIEW ATTRIBUTES command has been issued, the border is drawn with the highest possible color index for the surface on which the viewport appears.)

The border is always drawn in Replace mode (ALU mode 11) so that it overwrites the pixels below it. The border is removed by writing over the border, again in Replace mode, with the background wipe index for the viewport. Thus, turning the border on and off erases any pixels on the border of the viewport. (For a description of ALU mode 11, see the BEGIN PIXEL OPERATIONS command. For a description of the background wipe index, see SET VIEW ATTRIBUTES.)

This description of the "border-visibility-mode" parameter replaces the one in the right column of p.5-64:

border-visibility-mode: integer (keyword in Setup syntax); specifies whether the border of the current view is visible or invisible. Must be chosen from the following:

Host	Setup	
0	no	Invisible
1	yes	Visible
2	toggle	Toggles the border visibility (if visible, it becomes invisible; if invisible, it becomes visible)

Defaults: Factory = 0
Power-Up = 0
Omitted = 0

SET BREAK TIME

Sets the duration (in milliseconds) of the break signal the terminal sends when you press the Break key.

Host Syntax

```
EcNK break-time
```

Setup Syntax

```
BREAKTIME break-time
```

breaktime: integer; indicates the length of the break signal in milliseconds; must be in the range 0 through 65535. A value of 0 causes no break signal to be sent.

Defaults:	Factory	= 200
	Power-Up	= Saved in memory
	Omitted	= 0

SET BYPASS CANCEL CHARACTER

Specifies the character that causes the terminal to leave Bypass mode.

Host Syntax

```
EcNU bypass-cancel-character
```

Setup Syntax

```
BYPASSCANCEL bypass-cancel-character
```

bypass-cancel-character: small integer (in the range from 0 through 127); indicates the ADE value of the character that cancels Bypass mode; in Setup syntax, you can specify an ADE value or enter the corresponding ASCII character.

Defaults:	Factory	= 10 (LF)
	Power-Up	= Saved in memory
	Omitted	= 0 (do not go into Bypass mode)

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.

In Setup mode, to specify an ADE value in the range 0 through 9, precede the digit with 0. For example, indicate HT (ADE 9) as **09**. The single digits **0** through **9** indicate the digit characters (ADE 48 through ADE 57).

See the *Communications* section for an explanation of the terminal's Bypass mode.

SET CHARACTER PATH

Specifies the direction to move after writing each graphtext character.

Host Syntax

```
EcMN direction
```

Setup Syntax

```
GTPATH direction
```

direction: integer (keyword in Setup syntax); indicates which direction graphtext characters are written. Table 5-11 defines each value's effect.

Default: Factory = 0 (right)
Power-Up = 0 (right)
Omitted = 0 (right)

Table 5-11

CHARACTER PATH SETTINGS

Integer	Setup Keyword	Meaning
0	right	Use path equal to rotation angle
1	left	Use path 180° greater than rotation angle
2	up	Use path 90° greater than rotation angle
3	down	Use path 90° less than rotation angle

Figure 5-10 shows how the string "ABC" is displayed using the four different directions for character path. Note that graphtext rotation is 0° for all examples in Figure 5-10.

The keywords (**right, up, left, and down**) refer to character path *after* orientation with the rotation angle specified in SET GRAPHTEXT ROTATION. Table 5-11 gives explicit definitions of the keywords.

Integer Setting	Setup Parameter	Example	Path Relation to Rotation Angle
0	RIGHT	xABC*	Same
1	LEFT	*CBAx	+180°
2	UP	* C B A x	+90°
3	DOWN	x A B C *	−90°

Note: Graphtext is "ABC" with rotation = 0° in all examples.
x is graphics position before graphtext is displayed.
*** is updated graphics position after graphtext is displayed.**

4526-17B

Figure 5-10. Character Path Settings.

SET COLOR MODE

Sets the color mode for the terminal.

Host Syntax

```
EcTM color-specifying-mode
     color-overlay-mode
     gray-mode
```

Setup Syntax

```
CMODE color-specifying-mode
      color-overlay-mode
      gray-mode
```

color-specifying-mode: integer; specifies the color coordinate system used to mix colors in subsequent color operations. Valid values are:

0	No change
1	RGB (red, green, blue)
2	CMY (cyan, magenta, yellow)
3	HLS (hue, lightness, saturation)

Defaults: Factory = 3 (HLS)
Power-Up = 3
Omitted = 0 (Or no change)

color-overlay-mode: integer; specifies the mode used when colors are placed on top of each other. Valid values are:

0	No change
1	Opaque
2	Subtractive
3	Additive

Defaults: Factory = 1 (Opaque)
Power-Up = 1
Omitted = 0 (Or no change)

gray-mode: integer; specifies whether operation is color or black and white. Valid values are:

0	No change
1	Normal color operation

Defaults: Factory = 1 (Color mode)
Power-Up = 1 (Color mode)
Omitted = 0 (Or no change)

This command sets color mode parameters for the terminal. A color raster terminal generates colors on the display screen by mixing the three primary colors: red, green, and blue. However, the terminal allows the user to specify colors in one of three color systems: *HLS* (hue, lightness, and saturation); *RBG* (red, green, and blue); or *CMY* (cyan, magenta, and yellow). The SET COLOR MODE command selects one of these coordinate models.

HLS System. *Hue* is the angle formed by rotating a vector around a 360 degree color axis as shown in the *HLS System Color Cone*. Starting with a hue of 0 °, and rotating in a counterclockwise direction, 0 ° corresponds to blue, 120 ° to red and 240 ° to green, with intermediate shades corresponding to intermediate rotations.

Lightness is the position of a vector along the axis of the cone. A lightness of 0% is black with dark shades of gray increasing toward lighter tones until white is reached at 100%.

Saturation is represented by a radial vector formed perpendicular to the cone axis. For example, one could specify *saturation* for a hue of 0 ° (blue) and Lightness 50% (middle gray) by starting at a saturation varing from 0 to 100%. 0% would produce a light blue gray, then pass thru a medium blue at 50% ending up at dark blue for a final value of 100%.

RGB System. This system, which is also called the additive color system, defines colors as mixtures of three additive color primaries: red, green, and blue. You control the overall brightness of the color with the brightness of the primaries you use. You specify colors in the RBG system as integer percentages from 0% to 100%. For example, adding RGB as (100,0,100) produces magenta.

CMY System. This system, which is also called the subtractive color system, defines colors as mixtures of the three subtractive primaries: cyan, magenta, and yellow. In the CMY system, adding a greater percentage of a primary reduces the lightness of the displayed color. You specify colors in the CMY system as integer percentages from 0% to 100%. For example, combining CMY as (100,0,100) produces the color green.

Color Overlay Mode. The *color-overlay-mode* parameter specifies the behavior of the terminal's writing surfaces. If this parameter is zero, the color overlay mode is left unchanged.

In Opaque mode (parameter of 1), pictures drawn on a surface obscure pictures drawn on surfaces behind them. When the terminal is turned on, it powers up in Opaque mode.

In Subtractive mode (parameter of 2), pictures are drawn as if using transparent inks. The terminal behaves like a "light table," in which transparent overlays are placed on top of a diffusing light source.

NOTE

If you specify the SUBTRACTIVE color-overlay-mode in the SET COLOR MODE command, then you should also specify a background color of white (or some other light color) with the SET BACKGROUND COLOR command.

In Additive mode (parameter of 3), the images drawn on different surfaces act as if their colored inks were comprised of many small point light sources. Where colors on one surface overlap with colors on another surface, the light from the two surfaces' light sources combine. For instance, a red object on one surface and a green object on another surface would combine to produce a yellow color where the two objects overlap.

SET COPY SIZE

Sets the copy size.

Host Syntax

```
EcQA size
```

Setup Syntax

```
HCSIZE size
```

size: integer; use one of the following:

0	Selects default copy size (8½x11″)
1	Selects smaller copy size

Defaults: Factory = 0
Power-Up = 0
Omitted = 0

If you select the smaller size for a screen copy, the copy is one-half the default size. However, if you select the smaller size for a dialog area copy, the copy is smaller than the default size but larger than one-half the default size. Refer to the HARDCOPY command for additional information about screen and dialog area copies.

The smaller size produces a faster copy, but only in eight colors: black, white, red, green, blue, cyan, magenta, and yellow.

The smaller copy size also allows you to copy 132 columns on the same line. If, however, the default copy size is chosen and Column mode is set to 132, the extra 52 columns are wrapped to the next line.

If you are using a monochrome copier, selecting the smaller copy size does not produce a smaller copy.

SET CURRENT MATCHING CLASS

Establishes the inclusion and exclusion sets used in matching operations.

Host Syntax

```
EcSL  inclusion-set
      exclusion-set
```

Setup Syntax

```
SGMATCHINGCLASS  inclusion-set
                 exclusion-set
```

inclusion-set: integer array; specifies the set of classes used in the inclusion part of a matching operation. Valid integer values are:

−1	All attributes
1 — 64	A specific attribute

Defaults: Factory = Empty array
Power-Up = Empty array
Omitted = Empty array

exclusion-set: integer array; specifies the set of classes used in the exclusion part of a matching operation. Valid values are:

−1	All attributes
1 — 64	A specific attribute

Defaults: Factory = Empty array
Power-Up = Empty array
Omitted = Empty array

A matching operation is done for each defined segment (except segment 0, the crosshair cursor) when Segment −3 is specified in a segment command.

The matching operation works this way:

IF The intersection of a segment's class set with the inclusion set equals the inclusion set

AND The intersection of a segment's class set with the exclusion set equals the empty set

THEN The command is performed on the segment.

Otherwise, the command is not performed on the segment.

The elements in the *inclusion-set* and *exclusion-set* parameters are classes that the user includes or excludes in the current matching class. There are 64 possible classes (1 through 64), and all combinations are valid. If you assign −1 as a member of either set, that set includes all classes.

See the discussion of segment classes in the section titled *The Graphics Terminal.*

SET DIALOG AREA BUFFER SIZE

Specifies the maximum number lines of text stored in the dialog area buffer.

Host Syntax

```
EcLB  number-of-lines
```

Setup Syntax

```
DABUFFER  number-of-lines
```

number-of-lines: integer; indicates the maximum number lines in the dialog buffer; must be in the range 2 though 32767.

Defaults: Factory = 49
Power-Up = Saved in memory
Omitted = Error LB11

This command takes effect as soon as it is given. The dialog area buffer is changed to the size specified, and the cursor returns to the home position in the dialog area. For a more detailed description of the dialog area, see *The Graphics Terminal* section of this manual.

If you set size of the dialog buffer to less than the size of the dialog area, the size of the dialog area is automatically changed to match the size of the of dialog buffer.

The Ansi mode commands TEKOM and TEKSTBM can set a nonscrolling dialog area. When this is the case, if you specicfy a dialog buffer larger than the screen size (32 lines), the top and bottom margins are reset to lines 1 and 32. If Origin mode is Absolute and you specify a dialog buffer larger than the screen size, the Origin mode is set to Relative.

SET DIALOG AREA COLOR MAP

Specifies the color assigned to one or more color indices in the dialog area.

Host Syntax

```
EcTF color-mixture . . .
```

Setup Syntax

```
DACMAP color-mixture . . .
```

color-mixture: integer array; consists of groups of four integers (*quadruples*). Each quadruple specifies a color index (valid range 0 through 7) and the HLS coordinates of the color being assigned to that index: hue (a valid range of –32768 through 32767), lightness (a valid range of 0 through 100), and saturation (a valid range of 0 through 100). Unlike other integer arrays in Setup syntax, the first element must specify the number of following elements.

Defaults:		
Factory	=	See Table 5-12
Power-Up	=	Saved in memory
Omitted	=	No change to color map

When specifying a color in most commands, you use a color index, which is an integer in the range 0 through 7. SET DIALOG AREA COLOR MAP defines the color for one or more dialog area color indices. The graphics area has its own set of color indices that are defined with the SET SURFACE COLOR MAP.

The color assigned to Index 0 applies only to alphatext characters. For the dialog area background and character backgrounds, Index 0 always means "transparent." The concept of transparency is explained in *The Graphics Terminal* section.

For example, the following command specifies white (HLS coordinates 0, 100, 0) for Index 3 in the dialog area:

```
EcTF430F40
```

In Setup syntax, this command is:

```
DACMAP 4 3 0 100 0
```

The **4** indicates that four values follow.

See *The Graphics Terminal* section and the *Tektronix Color Standard* (in the appendices) for more details of how the terminal displays colors.

Table 5-12 shows the factory default colors assigned to color indices when the terminal is shipped from the factory.

Table 5-12

FACTORY DEFAULT COLOR INDICES

Color Index	Color Mixture	Color Coordinates[a]								
		H	L	S	R	G	B	C	M	Y
0	BLACK	0	0	0	0	0	0	100	100	100
1	WHITE	100	100	100	100	100	100	0	0	0
2	RED	120	50	100	100	0	0	0	100	100
3	GREEN	240	50	100	0	100	0	100	0	100
4	BLUE	0	50	100	0	0	100	100	100	0
5	CYAN	300	50	100	0	100	100	100	0	0
6	MAGENTA	60	50	100	100	0	100	0	100	0
7	YELLOW	180	50	100	100	100	0	0	0	100

[a] H = hue, L = lightness, S = saturation
R = red, G = green, B = blue
C = cyan, M = magenta, Y = yellow

SET DIALOG AREA INDEX

Specifies the color index for alphatext characters, character-cell background, and dialog area background.

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; indicates the color index of the characters displayed in the dialog area; must be in the range 0 through 65535.

Defaults:	Factory	= 1
	Power-Up	= Saved in memory
	Omitted	= 0

character-background-index: integer; indicates the color index used for each character cell background; must be in the range 0 through 65535. Index 0 indicates transparency.

Defaults:	Factory	= 0
	Power-Up	= Saved in memory
	Omitted	= 0

dialog-background-index: integer; indicates the color index of the dialog area background. Must be in the range 0 through 65535. This is the color of the dialog area before characters are written on it and after it is erased. Index 0 indicates transparency.

Defaults:	Factory	= 0
	Power-Up	= Saved in memory
	Omitted	= 0

Color Indices 0 through 7 represent colors identified by the dialog area's color indices. These colors are defined by SET DIALOG AREA COLOR MAP. The dialog area has its own indices separate from those used for graphics. When you specify a value greater than 7 for any color index, the terminal uses Index 7.

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. The color of a character cell is set by *character-background-index*.

When Color Index 0 is used as the *character-index*, it represents an opaque color just like Color Indices 1 through 7. However, when Color Index 0 is used as the *character-background-index* or the *dialog-background-index*, it means "make this area transparent." Graphics behind the dialog area show through a transparent area; if the character background is transparent, the dialog appears as if it is written on a piece of glass in fronts.

SET DIALOG AREA LINES

Specifies the maximum number of lines visible in the dialog area.

Host Syntax

```
EcLL number-of-lines
```

Setup Syntax

```
DALINES number-of-lines
```

number-of-lines: integer; sets the maximum size in lines for the dialog area; must be in the range 2 through 32.

Defaults	Factory	= 32
	Power-Up	= Saved in memory
	Omitted	= Error

If you set the number of dialog area lines to be greater than the size of the dialog buffer, the size of the dialog buffer is changed to be the same size as the dialog area.

If Column mode is set to 132, the maximum number of rows is 30, instead of 32.

SET DIALOG AREA VISIBILITY

Specifies whether the dialog area is visible.

Host Syntax

```
EcLV visibility-mode
```

Setup Syntax

```
DAVISIBILITY visibility-mode
```

visibility-mode: integer (keyword in Setup syntax) one of the following sets the dialog area to become either invisible or visible in Host or Setup modes.

Host	Setup	
0	no	Dialog area invisible
1	yes	Dialog area visible

Defaults:	Factory	= 1 (yes)
	Power-Up	= Saved in memory
	Omitted	= 1 (yes)

This command serves the same purpose as the Dialog key — it sets whether the dialog area is visible.

SET DIALOG VISIBILITY determines whether the dialog area is visible. ENABLE DIALOG AREA determines whether alphatext is sent to the dialog buffer. If the dialog area is enabled but not visible, alphatext is saved in the dialog buffer even though it is not visible. When SET DIALOG VISIBILITY is given (specifying 1 or yes), the alphatext in the dialog area becomes visible. The terminal will automatically scroll the dialog area, if necessary, to put the cursor in view.

SET DIALOG AREA WRITING MODE

Specifies whether the SP and _ (Underscore) characters overstrike or replace other characters in the dialog area.

Host Syntax

```
EcLM writing-mode
```

Setup Syntax

```
DAMODE writing-mode
```

writing-mode: integer (keyword in Setup syntax); 0 or **replace** indicates characters are replaced; 1 or **overstrike** indicates characters are overwritten.

Defaults:	Factory	= 0 (replace)
	Power-Up	= Saved in memory
	Omitted	= 0 (replace)

This command affects only how the Space and Underscore characters are displayed in the dialog area. If the dialog area is in Overstrike mode, you can underline alphanumeric characters.

Alphatext displayed in the graphics area is not affected by this command. (Alphatext is displayed in the graphics area when the dialog area is disabled.) SET GRAPHICS AREA WRITING MODE sets overstrike capability for graphics area alphatext.

SET DIALOG HARDCOPY ATTRIBUTES

Specifies the number of pages to be copied, the starting page, and how formfeed is interpreted.

Host Syntax

```
ᴱcQL  number-of-pages
      page-origin
      ᶠF-interpretation
```

Setup Syntax

```
HCDAATTRIBUTES  number-of-pages
                page-origin
                ᶠF-interpretation
```

number-of-pages: integer; determines the number of pages to be copied from the origin to the bottom of the dialog buffer. Must be in the range of 0 to 32767.

Defaults:	Factory	= 1
	Power-Up	= 1
	Omitted	= 1

page-origin: integer; determines the starting page for the copy. Must be one of the following:

0	Specifies the first visible line in the scroll
1	Specifies the top of the scroll
2	Specifies the bottom of the scroll

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

$^F{}_F$-interpretation: integer; determines how the terminal interprets ASCII formfeed in the dialog buffer. Must be one of the following:

0	Ignores $^F{}_F$ and divides buffer into 66 line pages (60 text lines plus 3 blank lines top and bottom)
1	Starts new page every 66 lines *or* when $^F{}_F$ appears in the text
2	Starts new page only when $^F{}_F$ appears in the text

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

If the origin is set to the bottom of the dialog buffer, the copy begins that many pages up from the bottom of the dialog buffer. If *number-of-pages* is 0, then there is no change from the last setting. If you specify a greater number of pages than the dialog buffer contains, only the contents of the dialog buffer are copied.

For pages that have lines longer than 80 characters, the terminal sends the extra characters without an intervening $^C{}_R$. This allows printers with widths greater than 80 columns to be used. The TEKTRONIX 4695 Color Copier automatically generates a $^C{}_R{}^L{}_F$ and starts printing on the next line for lines containing more than 80 characters. The 4695 prints the 132 characters on the same line if the SET COPY SIZE command has been executed with a parameter of 1 (to set the small copy size).

When using this command, you need to be aware of how *number-of-pages* and *$^F{}_F$-interpretation* interact with each other. The amount of text that goes on each page can vary depending on how the terminal divides the dialog buffer into pages, and that depends on the setting of *$^F{}_F$-interpretation*. If you issue this command with its default parameters, it copies 60 lines of text, beginning with the first visible line on the screen. If, however, you set *number-of-pages* to 4, *$^F{}_F$-interpretation* to 2, and have four $^F{}_F$'s beginning the text on the screen, then the copier turns out four blank sheets of paper. Additionally, if *number-of-pages* is 1, *page-origin* is 1, *$^F{}_F$-interpretation* is 2, and there are no $^F{}_F$'s in the text, then the copier prints the entire dialog buffer with no page breaks. (On the TEKTRONIX 4695 Copier, this would be a continuous sheet of paper).

You can stop a hardcopy operation by pressing the CANCEL key.

SET ECHO

Specifies whether the terminal echoes characters it transmits to the host.

Host Syntax

```
EcKE  echo-mode
```

Setup Syntax

```
ECHO  echo-mode
```

echo-mode: integer (keyword in Setup syntax); must be one of the following:

Host	Setup	
0	no	Sets remote echo; the terminal does not provide an echo
1	yes	Sets local echo; the terminal provides the echo

Defaults: Factory = 0 (no)
Power-Up = Saved in memory
Omitted = 1 (yes)

When a character is typed on the keyboard, the character displayed on the screen is an echoed character. Some computers provide an echo, and some do not. 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. If the host does provide the echo, set the terminal to remote echo; otherwise, characters typed on the keyboard will be displayed twice, LLIIKKEE TTHHIISS.

In Setup mode the terminal always provides the echo.

SET EDIT CHARACTERS

Host Syntax

```
EcKZ  character-delete
      line-delete
      take-literally
```

Setup Syntax

```
EDITCHARS  character-delete
           line-delete
           take-literally
```

character-delete: small integer; specifies which key will delete individual characters in Setup mode.
Defaults: Factory = 127 (DT)
Power-Up = Saved in memory
Omitted = Unchanged

line-delete: small integer; specifies which key will delete a line in Setup mode.
Defaults: Factory = 24 (CN)
Power-Up = Saved in memory
Omitted = Unchanged

take-literally: small integer; specifies character used in arrays to mean that "the following characters are to be taken literally, rather tha considered part of the array."
Defaults: Factory = 126 (~)
Power-Up = Saved in memory
Omitted = Unchanged

This version of the SET EDIT CHARACTERS command replaces the one on page 5-74.

SET EDIT CHARACTERS

Specifies the terminal's special editing characters used in Setup mode.

Host Syntax

$^{E}{}_{C}$**KZ** character-delete,
line-delete,
literal

Setup Syntax

```
EDITCHARS character-delete,
          line-delete,
          literal
```

character-delete: integer (small integer in Setup syntax); specifies the key used in Setup mode to erase the character just left of the cursor.

Defaults:	Factory	= 127 ($^{D}{}_{T}$)
	Power-Up	= Saved in memory
	Omitted	= Unchanged

line-delete: integer (small integer in Setup syntax); specifies the key used in Setup mode to erase the current line.

Defaults:	Factory	= 24 ($^{C}{}_{N}$)
	Power-Up	= Saved in memory
	Omitted	= Unchanged

literal: integer (small integer in Setup syntax); specifies the character used just before an editing character to suspend its control action and print it as text. Only the character immediately following the *literal* character is affected. Use the *literal* character before $^{C}{}_{R}$ and the three special editing characters defined in this command.

Defaults:	Factory	= 126 (~)
	Power-Up	= Saved in memory
	Omitted	= Unchanged

In Setup syntax, precede ADEs less than ten with a 0. For example, 08 indicateds ADE 8, the Back Space character. The single digit 8 indicates the character 8, whose ADE is 56. Refer to the ASCII Code Chart appendix.

SET EOF STRING

Specifies the terminal's end-of-file string.

Host Syntax

```
EcNE EOF-string
```

Setup Syntax

```
EOFSTRING EOF-string
```

EOF-string: integer array of small integers (delimited string in Setup syntax). Each integer in the array must be in the range from 0 through 127. The array must define no more than 10 ASCII characters.

Defaults:	Factory	= Empty array
	Power-Up	= Saved in memory
	Omitted	= Empty array

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

The end-of-file string should be set to match whatever string your host actually sends at the end of a file.

In Setup syntax you can enter either ASCII characters or ADEs for ASCII characters. Enter ADEs less than 10 with a leading zero. For example, to enter the Back Space character with ADE 8, enter **08**; just the character **8** will be interpreted as the numeral 8 with ADE 56.

SET EOL STRING

Specifies the terminal's end-of-line string.

Host Syntax

```
EcNT EOL-string
```

Setup Syntax

```
EOLSTRING EOL-string
```

EOL-string: integer array of small integers (delimited string in Setup syntax); each integer must be in the range from 0 through 127. This parameter defines a string of no more than two ASCII characters.

Defaults:	Factory	= 13 (CR)
	Power-Up	= Saved in memory
	Omitted	= Empty array

The terminal terminates each line of a report sent to the host with the end-of-line string. Typically this string consists of the single character CR (ADE 13). See the individual report descriptions for details.

In Setup syntax you can enter either ASCII characters or ADEs of ASCII characters. Enter ADEs less than 10 with a leading zero. For example, to specify the Back Space character, enter **08**; entering **8** specifies the numeral 8 with ADE 56.

SET EOM CHARACTERS

Specifies the character(s) used to terminate messages.

Host Syntax

```
EcNC first-EOM-indicator
     second-EOM-indicator
```

Setup Syntax

```
EOMCHARS first-EOM-indicator
         second-EOM-indicator
```

first-EOM-indicator: small integer (in the range from 0 through 127); indicates ADE of first end-of-message character. In Setup mode, you can enter the character or its ADE.

Defaults: Factory = 13 (CR)
Power-Up = Saved in memory
Omitted = 0 (NL)

second-EOM-indicator: small integer (in the range from 0 through 127); indicates ADE of second end-of-message character. In Setup mode, you can enter the character or its ADE.

Defaults: Factory = 10 (LF)
Power-Up = Saved in memory
Omitted = 0 (NL)

This command sets the end-of-message or *turnaround* character(s) for text transmission from the terminal to the host. This character or pair of characters is used for controlling the flow of communications between the terminal and the host.

For example, in Prompt mode, when the terminal transmits the end-of-message character(s), it stops transmitting until it receives a prompt string from the host *and* the transmit delay has expired (see SET TRANSMIT DELAY command). Then it sends the first character of the next line of text.

If the terminal is not in Prompt mode, after sending the end-of-message character(s), it waits until the transmit delay has expired before sending further information.

SET ERROR THRESHOLD

Specifies the kinds of error messages the terminal displays.

Host Syntax

```
EcKT error-threshold-level
```

Setup Syntax

```
ERRORLEVEL error-threshold-level
```

error-threshold-level: integer; indicates a minimum error level displayed. Must be in the range 0 through 4:

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
Power-Up = 2
Omitted = 0

This command determines only the level of error messages that are displayed on the screen. Regardless of what level you set, the terminal records the eight most recent error messages and transmits them in response to a REPORT ERRORS command.

SET FIXUP LEVEL

Specifies how often to update the screen display when changes are made to the current view.

Host Syntax

```
EcRF  fixup-level
```

Setup Syntax

```
FIXUP  fixup-level
```

fixup-level: integer; specifies how frequently the terminal updates the current view in its display. Must be in the range 0 through 6.

Defaults:		
	Factory	= 6
	Power-Up	= 6
	Omitted	= 0

A positive fixup level not listed in the chart has the same effect as the next lower fixup level that is listed. For instance, Fixup Level 3 has the same effect as Fixup Level 2; Fixup Level 100 has the same effect as Fixup Level 6. A negative fixup level sets the fixup level to 0, and an error is generated.

The following chart lists the fixup levels and their meanings:

Fixup Level	**Meaning**
0	A RENEW VIEW command, a PAGE command, or pressing the G Erase or S Erase key completely updates the view. (At Fixup Level 0, these are the only times the viewport contents are altered.)
2	Performs the action for Fixup Level 0. Besides this, updates the display as additions are made to the current view. (For instance, when a line is drawn in the current view, the display is updated to show that line. And, when a segment is made visible, the display is updated to show that segment.) (At Fixup Level 2, when a segment is moved, it is drawn in the new position, but not erased from its old position. The display is not updated when a segment is made invisible or deleted.)
4	Performs the actions for Fixup Levels 0 and 2, and erases segments displayed in XOR mode (by drawing them again in XOR mode) in response to commands that change the segment's visiblity or position in terminal space. Segments displayed in Set mode are treated as in Fixup Level 2 (see SET SEGMENT WRITING MODE for a discussion of Set and XOR modes).
6	Performs the actions for Fixup Levels 0 through 4, and erases a segment whenever it is made invisible or is deleted from the viewport. (For segments drawn in Set mode, this is done by redrawing the segment in the current wipe index. For segments drawn in XOR mode, this is done by redrawing the segment in XOR mode.)

SET FLAGGING MODE

Specifies the kind of flagging the terminal uses.

Host Syntax

```
EcNF  flagging-mode
```

Setup Syntax

```
FLAGGING flagging-mode
```

flagging-mode: integer (keyword in Setup syntax); must be one of the following:

Host	Setup	
0	none	Specifies no flagging
1	input	Specifies $^D1/^D3$ flagging on input from the host
2	output	Specifies $^D1/^D3$ flagging on output to the host
3	in/out	Specifies $^D1/^D3$ flagging on both input from and output to the host
4	DTR/CTS	Specifies DTR (Data Terminal Ready) and CTS (Clear To Send) signal lines at the RS-232 connector to regulate data flow

Defaults: Factory = 0 (none)
Power-Up = Saved in memory
Omitted = 0 (none)

Flagging allows the terminal and host to signal each other when each is 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.

If the host uses the $^D1/^D3$ scheme, the operator can use the Ctrl-S and Ctrl-Q keys to stop and start output from the host. See the *Communications* section for more details.

SET GIN AREA

Allows a specified GIN area on an external GIN device to map onto the terminal's coordinate space.

Host Syntax

```
EcIV  device-function-code
      window-specifier
      GIN-lower-left-corner
      GIN-upper-right-corner
```

Setup Syntax

```
GINAREA device-function-code
        window-specifier
        GIN-lower-left-corner
        GIN-upper-right-corner
```

device-function-code: integer; specifies which device functions are to be affected by this area. See Table 5-6 (under ENABLE GIN) for valid device-function codes.
Defaults: Factory = All
Power-Up = all
Omitted = 0

window-specifier: integer; specifies the terminal space window into which the GIN area maps. The valid range of values are:

−1	Window specified by SET GIN WINDOW command
0	Window of current view

Defaults: Factory = −1
Power-Up = −1
Omitted = 0

GIN-lower-left-corner: xy-coordinate; specifies the lower-left corner of a rectangular region in the GIN device space. The valid range for both x- and y-coordinates include 0 to 4095.
Defaults: Factory = 0,0
Power-Up = 0,0
Omitted = 0,0

GIN-upper-right-corner: xy-coordinate; specifies the upper-right corner of a rectangular region in the GIN device space. The valid range of both x- and y-coordinates include 0 to 4095.

Defaults:	Factory	= 4095,4095
	Power-Up	= 4095,4095
	Omitted	= 4095,4095

This command maps a specified rectangular region on a GIN device (a GIN area) onto a specified rectangular window in terminal coordinate space.

Device-Function. When you assign a GIN area to a device-function, the GIN area is assigned to all functions of the specified device. For example, if you assign a GIN area to the tablet-Locator device-function, and then invoke the tablet-Pick device-function, the same GIN area is used.

Window Specifier. If you specify a *window-specifier* of −1, the GIN device area (specified by the third and fourth parameters) is mapped into the terminal space region specified by the most recent SET GIN WINDOW command. The specified GIN device area remains mapped into the specified terminal space window until another SET GIN AREA command remaps it by enclosing the former device area within its own.

If you specify a *window-specifier* of 0, the GIN device area is mapped into the window of the current view each time the device is enabled and a point within the GIN area is selected. This window changes when you change views. The window also changes when an operator does a Zoom or Pan operation.

Coordinates. This command assumes that the GIN device is 4096-by-4096 units square. If the device is rectangular, the command assigns 4096 units to the long axis of the device, and an proportionate number of units to the short axis.

The x- and y- coordinates you assign are sorted so that any two corners specify the rectangle.

If you specify the two x-coordinates the same or the two y-coordinates the same (if the GIN area is zero in either direction) then an error is generated, except when you specify both corners to be (0,0), in which case, the GIN area defaults to from (0,0) to (4095,4095).

You can define multiple GIN areas for the tablet but only one for joydisk or delta mode tablet. If the device specified in the *device-function* parameter is the terminal joydisk or delta mode tablet, the GIN area must be from (0,0) to (4095,4095). If you define overlapping areas on non-joydisk devices, and the device position is inside more than one GIN area, the most recently defined GIN area and *window-specifier* values are used.

GIN coordinate data for the device is mapped from the GIN area on the associated window. If the window is that of the current view (*window-specifier* = 0), the movement rate remains in proper scale to the window, regardless of the zoom scale factor.

To delete a GIN area, define a GIN area which totally covers the GIN area you want to delete. To delete all GIN areas, define both the GIN window and the GIN area to be from (0,0) to (4095,4095).

SET GIN CURSOR

Selects a segment for use as the cursor for a particular device function code.

Host Syntax

```
EcIC device-function-code
     segment-number
```

Setup Syntax

```
GINCURSOR device-function-code
          segment-number
```

device-function-code: integer; enables the specified GIN device and its function. For valid values, see Table 5-6 under ENABLE GIN.
Defaults: Omitted = Error IC20

segment-number: integer; selects a specific segment in the range 0 to 32767 as a graphics cursor. Segment value 0 is the crosshair cursor.

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= Error IC20

SET GIN CURSOR specifies which segment is to be used as the graphics cursor for all subsequent operations, using the specified graphics input device-function combination. No action is seen when this command is given; the result will be seen later, when an ENABLE GIN command is issued for the specified device-function combination.

When a segment is enabled as the GIN cursor, it is the segment's pivot point which replaces the intersection of the crosshairs in the default crosshair cursor.

For selecting the GIN cursor, the ENABLE 4010 GIN command is the same as an ENABLE GIN command for *device-function-code* 0 (thumbwheels device, Locator function). That is, a SET GIN CURSOR command for *device-function-code* 0 selects a cursor not only for use with ENABLE GIN commands with *device-function-code* 0, but also for use with ENABLE 4010 GIN commands.

When the named device-function is later enabled, the segment attributes of this segment are changed as follows:

- Detectability is turned off.
- Visibility is turned on. (If the sampled device is the tablet, the segment is visible when the pen is in presence; when out of presence its visibility depends on the segment's original visibility attribute.)

As the sampled device is manipulated, the cursor segment's position is continuously updated. When a GIN event occurs, the cursor position is sent to the host as part of the GIN report for that event.

When the device is disabled, the display mode, detectability, and visibility attributes are restored to their original values. The segment's postion attribute, however, is not restored.

Defaults. If no SET GIN CURSOR command has been issued, Segment 0 — the standard crosshair cursor — serves as the graphics cursor. By use of the SET GIN CURSOR command, however, another segment may be used as a graphics cursor.

If Setup mode is entered while GIN is enabled, the graphics cursor will disappear until Setup mode is exited. While in Setup mode, the enabled device (joydisk or tablet) will *not* be used for input to the suspended GIN operation.

Several device-functions can specify the same cursor although the cursor movement may be undefined if an attempt is made to use more than one such device-function at a time.

If the specified segment does not exist, or if this command is issued while the specified function is enabled, an error occurs.

The cursor is scaled according to the current window viewport transform and segment image transform. The pivot point of the cursor cannot be moved outside of the current viewport. If another view is selected as the current view, the cursor moves to that view's viewport and is scaled to reflect the new view's window viewport transform. However, the default crosshair cursor, unlike other segments, is not clipped at the viewport boundary.

If the cursor is the default cursor (Segment 0, the crosshair cursor), its position may be set with the SET SEGMENT POSITION command. However, the default cursor cannot be positioned outside the window for the current view. The terminal will position it as close to the specified point as possible, constraining it to be within the current window. Any graphic input device motion will restore the default cursor to its position prior to the SET SEGMENT POSITION command.

If a cursor other than the default cursor is used, its position may be set using either the SET SEGMENT POSITION command or the SET SEGMENT IMAGE TRANSFORM command. If the nondefault cursor is set outside the current window, it ceases to be visible in the current viewport. However, any graphics input device motion will restore the cursor back to its position prior to the SET SEGMENT POSITION or SET SEGMENT IMAGE TRANSFORM command. It will then move from that point in response to the graphics input device. This device may not, by itself, move the cursor outside the window.

SET GIN CURSOR COLOR

Specifies the color mixture for the graphics crosshair cursor.

Host Syntax

```
ᴱcTC 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 color value for the GIN crosshair cursor, arguments are interpreted according to current color mode. (See SET COLOR MODE command and Table 5-13.)

Defaults: Factory = 0
Power-Up = Saved in memory
Omitted = 0

second-color-coordinate: integer; selects a color value for the GIN crosshair cursor, arguments are interpreted according to current color mode. (See SET COLOR MODE command and Table 5-13.)

Defaults: Factory = 100
Power-Up = Saved in memory
Omitted = 0

third-color-coordinate: integer; selects a color value for the GIN crosshair cursor, arguments are interpreted according to current color mode. (See SET COLOR MODE command and Table 5-13.)

Defaults: Factory = 0
Power-Up = Saved in memory
Omitted = 0

Table 5-13

SET GIN CURSOR COLOR PARAMETER VALUES

Parameter	HLS	RGB	CMY
first-color-coordinate	0 — 360° (Hue)	0 — 100 (Red)	0 — 100 (Cyan)
second-color-coordinate	0 — 100 (Lightness)	0 — 100 (Green)	0 — 100 (Magenta)
third-color-coordinate	0 — 100 (Saturation)	0 — 100 (Blue)	0 — 100 (Yellow)

This command specifies the color mixture for the graphics cursor according to the current color coordinate model (HLS, RGB, CMY). This color is independent of the eight dialog indices and the eight graphics indices and may be any of the 64 colors.

SET GIN DISPLAY START POINT

Specifies an initial point for GIN inking or GIN rubberbanding.

Host Syntax

```
EcIX device-function-code
     start-point
```

Setup Syntax

```
GINSTARTPOINT device-function-code
              start-point
```

device-function-code: integer; enables the specified GIN device and its function. For valid values, see Table 5-6 (under ENABLE GIN).
Defaults: Omitted = 0

start-point: xy-coordinate; specifies the initial point in terminal space coordinates to start an ink line or a rubberband line. The valid range of both xy-coordinates include 0 to 4095.
Defaults: Omitted = 0,0

This command sets the *start-point* that is used by the SET GIN RUBBERBANDING and SET GIN INKING commands when they are enabled with a parameter of 2.

SET GIN GRIDDING

Restricts the set of possible GIN positions for Locate or Pick functions.

Host Syntax

```
EcIG device-function-code
     x-grid-spacing
     y-grid-spacing
```

Setup Syntax

```
GINGRIDDING device-function-code
            x-grid-spacing
            y-grid-spacing
```

device-function-code: integer; enables the specified GIN device and its function. For valid values, see Table 5-6 (under ENABLE GIN).
Defaults: Omitted = 0

x-grid-spacing: integer; sets the spacing between invisible horizontal grid space. The valid range of values include 0 to 4095.

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

y-grid-spacing: integer; sets the spacing between invisible vertical grid space. The valid range of values include 0 to 4095.

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

This command causes gridding for all subsequent operations of the specified Locator or Pick functions. The x,y parameters specify an invisible grid that covers the entire terminal space. All further graphics cursor movement is constrained so that the cursor will always lie on points corresponding to the intersections of these grid lines.

An x-spacing (or y-spacing) of zero disables gridding in the x-direction (y-direction). Specifying zero for both these parameters turns off the gridding feature.

A grid spacing of one serves no useful purpose, since points cannot be closer together than one terminal space unit. (It would be better to specify a spacing of zero, and thus save some time by disabling gridding.) Note also that large grid spacings are impractical. X- and y-spacings of 4095 would permit only four accessible points, at the four corners of terminal space.

Gridding applies only to the Locator and Pick functions; gridding is not permitted for the Stroke function.

Default is no gridding (x- and y-spacing both zero).

For purposes of gridding, an ENABLE 4010 GIN command is considered to be an ENABLE GIN command for *device-function* code 0. That is, a SET GIN GRIDDING command for *device-function* 0 (joydisk device, Locator function) affects graphics input in response to an ENABLE 4010 GIN command as well as graphics input in response to an ENABLE GIN command for *device-function* zero.

SET GIN INKING

Turns inking on or off for all subsequent operations of the specified Locator or Stroke functions.

Host Syntax

```
ᴱcII device-function-code
     inking-mode
```

Setup Syntax

```
GININKING device-function-code
          inking-mode
```

device-function-code: integer; enables the specified GIN device and its function. For valid values, see Table 5-6 (under ENABLE GIN).
Defaults: Omitted = 0

inking-mode: integer; valid values are:

0	Disables the inking feature
1	Draws a line between the point selected by the preceding Locator event and the subsequent Locator event
2	Draws a line between the GIN display start-point and the subsequent Locator event

Subsequent lines are then constructed identical to *inking-mode* 1.

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

This command turns inking on or off for all subsequent operation of the specified Locator or Stroke function (inking is not allowed for the Pick function).

When inking is turned on, each Locator event after the first causes a line to be drawn to the point selected by the Locator event. During Stroke functions, a line is drawn between each point in the Stroke.

The line is drawn in Set mode using the current line style and line index.

If *inking-mode* is 2, the first GIN point is "inked" from the GIN display start-point, set by the SET GIN DISPLAY START POINT command. After the first point, it is identical to *inking-mode* = 1.

If *inking-mode* is 1, and *rubberbanding-mode* is set to 2 by the SET RUBBERBANDING MODE command, then GIN inking behaves as if *inking-mode* was set to 2.

You can use inking in Locate or Stroke functions so the operator can see where the GIN device has moved. When the operator causes a GIN event, the terminal draws a line from the location of the last GIN location to the current location. You can start inking after the first point is entered if you enable Inking mode with a value of 1, or start inking immediately from the GIN Display Start Point if you enable Inking mode 2.

SET GIN RUBBERBANDING

Turns rubberbanding on or off for all subsequent operations of the specified Locator function.

Host Syntax

```
EcIR device-function-code
     rubberbanding-mode
```

Setup Syntax

```
GINRUBBERBANDING device-function-code
                 rubberbanding-mode
```

device-function-code: integer; enables the specified GIN device and its function. For valid values, see Table 5-6 under ENABLE GIN.
Defaults: Omitted = 0

rubberbanding-mode: integer; valid values are:

0	Disables the rubberbanding feature
1	Causes rubberbanding to occur between the most recent GIN Locator event and the current cursor position
2	Causes initial rubberbanding to occur between the GIN Display Start Point and the current cursor position Subsequent rubberbanding is constructed identical to mode 1.

Defaults: Factory = 0
Power-Up = 0
Omitted = 0

This command turns rubberbanding on or off for all subsequent operations of the specified Locator function. Rubberbanding is not allowed for the Pick and Stroke functions.

With rubberbanding turned on and the specified *device-function* enabled, a line is drawn from the most recently selected point to the current cursor position. The line is drawn in XOR mode, using the current line style and line index. If dashed lines are used, the pattern of dashes begins at the fixed end of the line.

If *rubberbanding-mode* is 2, the first GIN point is "rubberbanded" from the *GIN-display-start-point*, set by the SET GIN DISPLAY START POINT command. After the first point, it is identical to *rubberbanding-mode* = 1.

If *inking-mode*, as set by the SET-INKING-MODE command, is 1, and *rubberbanding-mode* is 2, then GIN inking behaves as if *inking-mode* was set to 2.

If rubberbanding is turned on or off while the GIN device is enabled, only subsequent points are affected.

The beam position is not affected by rubberbanding.

You can use rubberbanding with the Locate function to draw an elastic line between the GIN start point or last GIN event and the location of the current cursor position. As with inking, you can control rubberbanding with parameters to the command. The terminal draws the elastic line in XOR mode, using the current line style and line index. If dashed lines are used, the portion of dashed lines begins at the fixed end of the line. If ruberbanding is turned on or off while the GIN device is enabled, only subsequent points are affected. The electron beam position is not affected by the rubberbanding mode.

SET GIN STROKE FILTERING

Restricts the number of GIN Stroke Reports sent to the host.

Host Syntax

```
EcIF device-function-code
     distance-filter
     time-filter
```

Setup Syntax

```
GINFILTERING device-function-code
             distance-filter
             time-filter
```

device-function-code: integer; enables the specified GIN device and its function. Only valid for device-function codes 10 and 18 (see Table 5-6 under ENABLE GIN).

Defaults:	Omitted	= Error IF10 (No Stroke filtering on joydisk)

distance-filter: integer; sets the distance the GIN device must be moved before issuing a GIN Stroke Report. 0 disables the distance filter. Valid values in the range 1 to 4095 establish the minimum distance in terminal space for either x or y movement to reach the distance threshold. Once exceeded, the terminal waits until the time threshold (if enabled) is exceeded before sending a GIN Stroke Report to the host.

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

time-filter: integer; sets the minimum interval between GIN Stroke Reports. 0 disables the time filter. Valid values in the range of 1 to 32767 establish the minimum interval in milliseconds (resolution ~ 33ms) between Stroke events. Once exceeded, the terminal waits until the distance threshold is exceeded before sending a GIN Stroke Report to the host.

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

The specified Stroke filtering parameters are applied to all subsequent operations of the specified Stroke function.

Time Filter. The *time-filter* parameter specifies the minimum time in milliseconds that will elapse between successive points. As the operator moves the tablet pen or four-button cursor, GIN stroke reports are sent to the host only at intervals of "time" milliseconds.

The terminal measures time in increments of about 33 milliseconds. Thus, specifying a time of 10 ms is, for practical purposes, the same as specifying a time of zero. Again, for pratical purposes, 27 ms is the same as 33 ms.

If both the *distance-filter* and *time-filter* parameters are non-zero, then all criteria of both parameters must be met for an Stroke report to be sent to the host.

When Stroke filtering is first enabled (or when the tablet stylus or cursor is lifted away from the tablet), the filters are reset so that at the first pen contact, a point is returned.

Note that if either the distance or the time parameter is zero, then that type of filtering is absent. With no filtering, points are output at the maximum speed of the tablet interface.

Filtering does not affect the cursor movement, but does affect the image formed by inking: inking occurs only between those points whose coordinates are reported to the host.

Locator and Pick functions are not affected. Default is no filtering.

When you are using the Stroke function with a graphics tablet, you may want to restrict the number of GIN Stroke events and reports that the terminal generates. You can specify (1) the minimum distance the tablet device moves before sending a report with a distance filter and (2) the minimum time between reports with a time filter. When both filters have values greater than 0, they act as thresholds and must be both exceeded before the terminal will send a GIN Stroke Report. When Stroke filtering is first enabled (or when the tablet stylus or cursor is lifted away from the tablet), the filters are reset so that at the first pen contact, a GIN Stroke Report is sent to the host.

SET GIN WINDOW

Sets the terminal space window to be used by the SET GIN AREA command with a parameter of −1.

Host Syntax

```
EcIW  lower-left-corner
      upper-right-corner
```

Setup Syntax

```
GINWINDOW  lower-left-corner
           upper-right-corner
```

lower-left-corner: xy-coordinate; specifies (in terminal space) one corner of the GIN window. Valid values for the x- and y-coordinates range from 0 to 4095.

Defaults:	Factory	= 0,0
	Power-Up	= 0,0
	Omitted	= 0,0

upper-right-corner: xy-coordinate; specifies (in terminal space) the opposite corner of the GIN window. Valid values for the x- and y-coordinates range from 0 to 4095.

Defaults:	Factory	= 4095,4095
	Power-Up	= 4095,4095
	Omitted	= 4095,4095

You can establish a rectangular area in terminal space as the GIN window. This window is used for the GIN area specified with the window specifier of −1.

This command sets a window in terminal space which can be used by the SET GIN AREA command. If SET GIN AREA specifies a *window-specifier* of −1, then the GIN area also specified by that command is mapped into the window in terminal space specified by the latest SET GIN WINDOW command.

The x- and y- coordinates you assign are sorted so that any two corners specify the rectangle.

SET GRAPHICS AREA WRITING MODE

Specifies whether the terminal overwrites or replaces a character in the graphics area of the display.

Host Syntax

```
EcMG  writing-mode
```

Setup Syntax

```
GAMODE  writing-mode
```

writing-mode: integer (keyword in Setup syntax); must be chosen from the following:

Host	Setup	
0	replace	Specifies replace
1	overstrike	Specifies overstrike

Defaults:	Factory	= 1 (overstrike)
	Power-Up	= Saved in memory
	Omitted	= 0 (replace)

This command affects the way alphatext (not displayed in the dialog area) and String-precision graphtext are displayed in the graphics area.

If *writing-mode* is set to overstrike, a character will be displayed on top of an existing character without erasing the existing character. You can use overstrike to underline characters with (the Underscore character).

If *writing-mode* is set to replace, the terminal erases the existing character (with the current wipe index specified by SET VIEW ATTRIBUTES) and displays the new character in place of the existing one.

Alphatext displayed in the dialog area is not affected by this command. (Alphatext is displayed in the dialog area when the dialog area is enabled.) SET DIALOG AREA WRITING MODE sets overstrike capability for dialog area alphatext.

SET GRAPHTEXT FONT

Determines which character font will be used to display subsequent stroke-precision graphtext.

Host Syntax

```
EcMF font-number
```

Setup Syntax

```
GTFONT font-number
```

font-number: integer; specifies the character font that will be used to display subsequent graphtext. Must be in the range 0 through 32767.

Defaults:	Factory and Power-Up	=	0 if ASCII keyboard 1 if Swedish keyboard 2 if German keyboard 3 if United Kingdom keyboard 9 if Danish/Norwegian keyboard 12 if French keyboard
	Omitted	=	0

Fonts are numbered from 0 through 32767; Fonts 0, 1, 2, 3, 9, and 12 are predefined as listed in Table 5-14. The terminal detects an error if you issue this command for an undefined font.

Table 5-14

PREDEFINED GRAPHTEXT FONTS

Font Number	Graphtext Font
0	Standard ASCII
1	Swedish
2	German
3	United Kingdom
9	Danish/Norwegian
12	French

Any undefined characters in a user-defined font default to the corresponding characters of the predefined font with the same font number. If there is no predefined font with that number, the corresponding characters of Font 0 are used.

SET GRAPHTEXT FONT GRID

Creates a graphtext font and specifies the dimensions of the grid (character cell) used for defining characters in the font.

Host Syntax

```
EcSG font-number
     grid-width
     grid-height
```

Setup Syntax

```
GTGRID font-number
       grid-width
       grid-height
```

font-number: integer; names the graphtext font for which a font grid is being defined. Must be in the range 0 to 32767.
Defaults: Omitted = 0

grid-width: integer; specifies the width of each grid in terminal space units. Must be in the range 1 to 4095.
Defaults: Omitted = Error

grid-height: integer; specifies the height of each grid in terminal space units. Must be in the range 1 to 4095.
Defaults: Omitted = Error

You can consider the grid to be a character cell; the grid width and height are those of an uppercase character. When defining a character with the BEGIN GRAPHTEXT CHARACTER command, the coordinates the terminal receives should stay generally within the bounds of the grid. However, characters can extend outside their font grids (as shown in Figure 5-11). The font grid is only used in scaling characters to fit the current graphtext size.

This command must be given before any characters are defined in the specified font. An error occurs, however, if any user-defined characters exist in the specified font. Once you delete all user-defined characters in that font (see DELETE GRAPHTEXT CHARACTER) you may issue a SET GRAPHTEXT FONT GRID command.

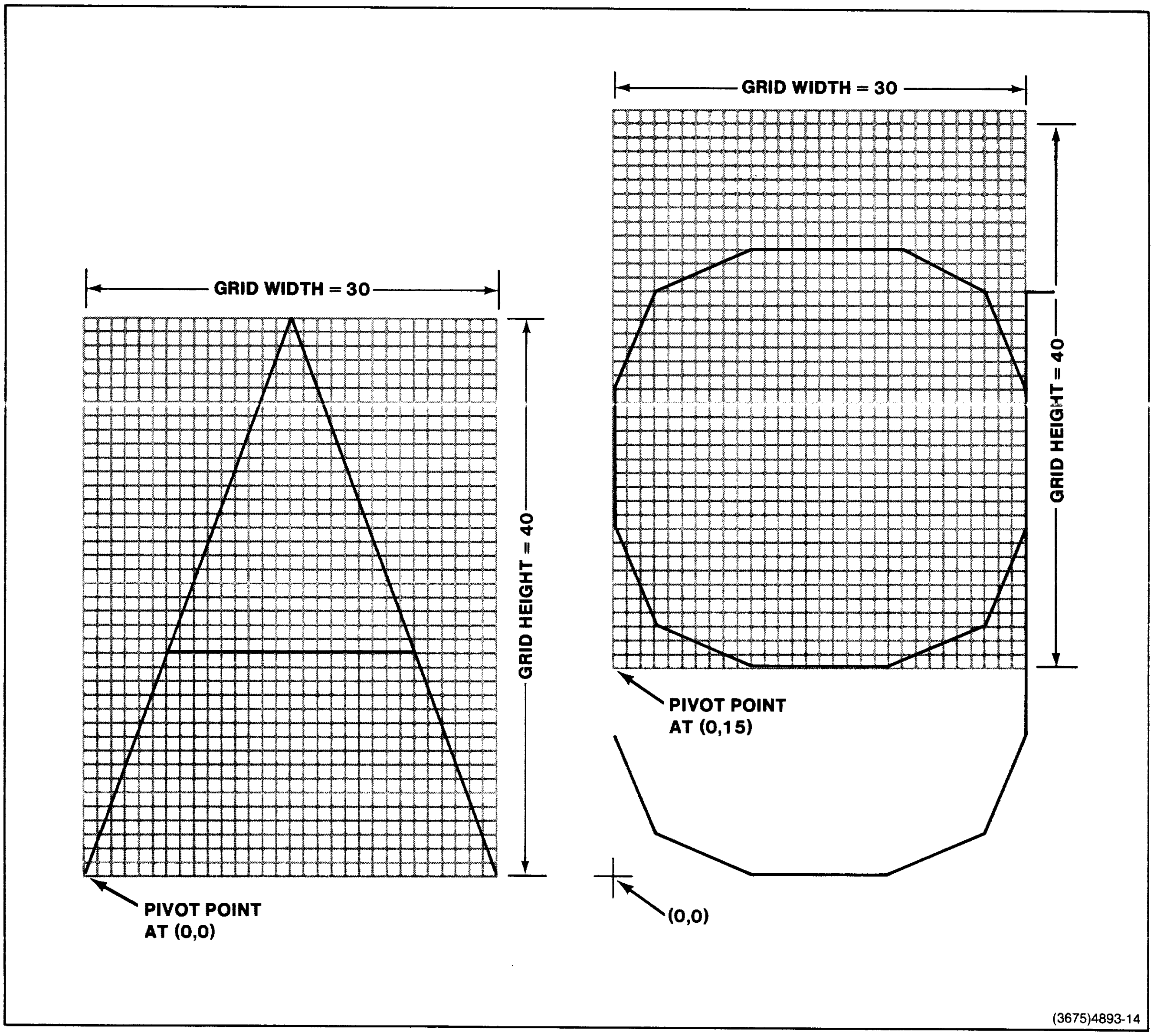


Figure 5-11. Font Grids for Two User-Defined Characters.

SET GRAPHTEXT PRECISION

Selects string-precision or stroke-precision to draw graphtext characters.

Host Syntax

```
EcMQ precision
```

Setup Syntax

```
GTPRECISION precision
```

precision: integer (keyword in Setup syntax). Must be chosen from the following:

Host	Setup	
1	string	Specifies string-precision
2	stroke	Specifies stroke-precision

Defaults: Factory = 2
Power-Up = 2
Omitted = Error

When string-precision is selected, the terminal displays graphtext using dot matrix characters similar to alphatext. Unlike alphatext, however, string-precision graphtext can be scaled with the SET GRAPHTEXT SIZE command, rotated with the SET GRAPHTEXT ROTATION command, and have its character path set with the SET CHARACTER PATH command. See the descriptions of these commands for detailed information.

When stroke-precision is selected, the terminal displays graphtext using Stroke characters. This permits more precise settings of the character size and rotation angles. Stroke-precision graphtext also permits a broader range of character styles because you can define characters to suit individual needs using the BEGIN GRAPHTEXT CHARACTER command.

SET GRAPHTEXT ROTATION

Specifies the rotation angle (in degrees) for all subsequent graphtext strings.

Host Syntax

```
EcMR angle
```

Setup Syntax

```
GTROTATION angle
```

angle: real; specifies the rotation angle in degrees, from the direction of the positive x-axis. Must be in the range −32767.0 to 32767.0.

Defaults:	Factory	= 0.0
	Power-Up	= 0.0
	Omitted	= 0.0

The rotation angle may be any real number in the range −32767.0 to 32767.0; the terminal converts this modulo 360 to a number in the range 0.0 to 360.0 degrees. Positive angles represent counterclockwise rotations, while negative angles represent clockwise rotations. An angle of 0 ° means that individual graphtext characters are displayed in their nomral, upright positions. Each subsequent character in a string is normally displayed to the right of the previous character; however, this can be changed by the SET CHARACTER PATH command.

For string-precision graphtext, the terminal arrays the text string along a line that is rotated to the specified rotation angle. However, the characters themselves only rotate to the nearest multiple of 90°. Thus, characters at angles between:

0.0 and 45.0°	=	0°
45.0 and 135.0°	=	90°
135.0 and 225.0°	=	180°
225.0 and 315.0°	=	270°
315.0 and 360.0°	=	0°

Stroke-precision graphtext can be displayed at any rotation angle. The characters in the text string rotate in concert with the line of text.

Figure 5-12 compares the stepped effect that occurs when rotating string-precision graphtext to the smooth rotation of stroke-precision graphtext.

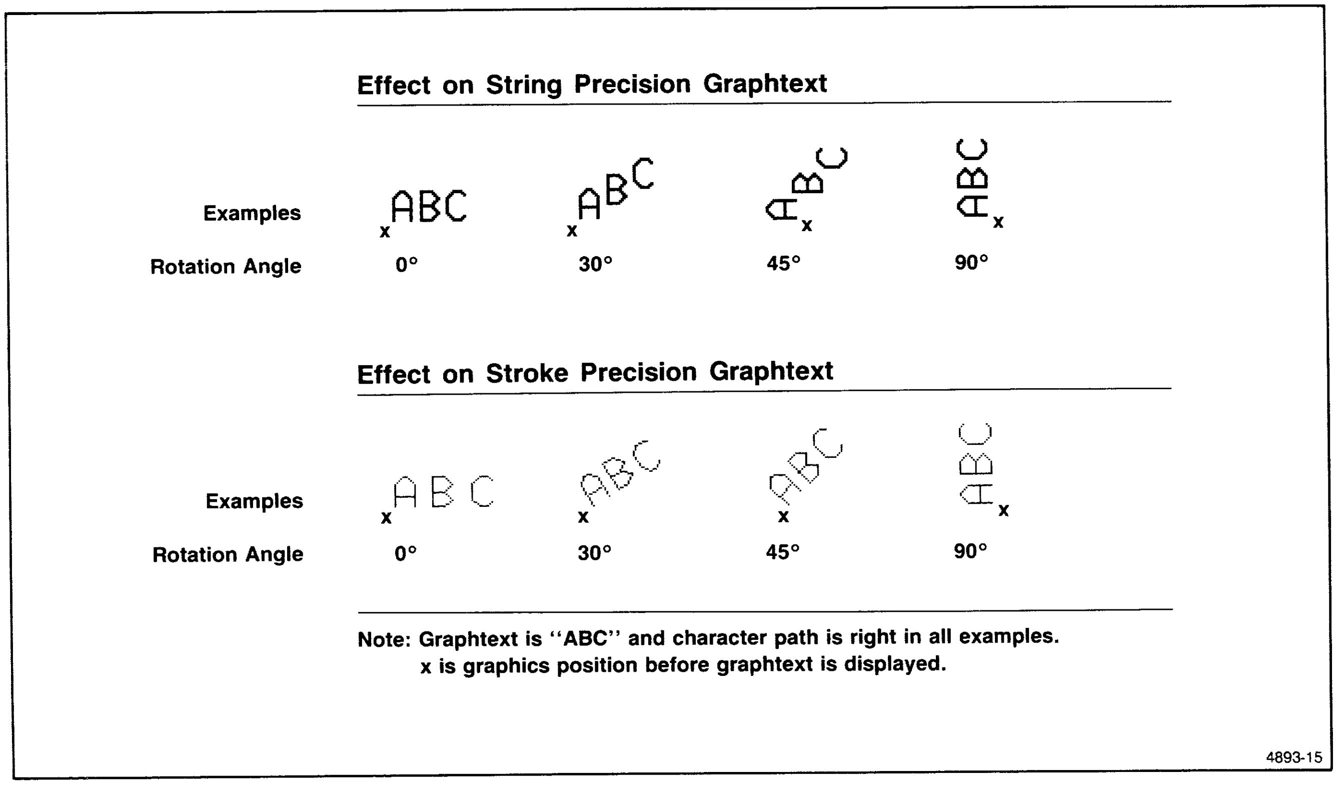

Effect on String Precision Graphtext
Examples
ABC
ABC
ABC
ABC
Rotation Angle
0°
30°
45°
90°
Effect on Stroke Precision Graphtext
Examples
ABC
ABC
ABC
ABC
Rotation Angle
0°
30°
45°
90°
Note: Graphtext is "ABC" and character path is right in all examples.
x is graphics position before graphtext is displayed.
4893-15

Figure 5-12. Graphtext Rotation Examples.

SET GRAPHTEXT SIZE

Sets the size of graphics text.

Host Syntax

```
EcMC width
     height
     spacing
```

Setup Syntax

```
GTSIZE width
       height
       spacing
```

width: integer; specifies width of a graphtext character cell, in terminal space units. Must be in the range 0 through 4095; 0 specifies the default values.

Defaults:	Factory	= 39
	Power-Up	= 39
	Omitted	= Error

height: integer; specifies height of a graphtext character cell, in terminal space units. Must be in the range 0 through 4095; 0 specifies the default values.

Defaults:	Factory	= 59
	Power-Up	= 59
	Omitted	= Error

spacing: integer; specifies the spacing, in terminal space units, between adjacent character cells in the same graphtext string. Must be in the range 0 through 4095.

Defaults:	Factory	= 12
	Power-Up	= 12
	Omitted	= 0

When displaying stroke-precision graphtext, the *width* and *height* define the size of a character. The *spacing* determines the size of the space between one character cell and the next. Where the space occurs depends on the character path set by the most recent SET CHARACTER PATH command. For instance, if the character path is "right", the space occurs between the right edge of one character cell and the left edge of the next. However, if the character path is "down", the space occurs between the bottom edge of one character cell and the top edge of the next.

When displaying string-precision graphtext, the terminal uses only the height parameter; the width and spacing parameters are accepted and stored, but ignored. The terminal displays string precision graphtext in several fixed sizes. The smallest available size displays an uppercase letter as seven pixels wide and nine pixels high; the other available sizes are integer multiples of the smallest size. Table 5-15 gives the height ranges that yield the first three character sizes available. The terminal examines the height you specify and displays the string-precision graphtext in the closest available size.

Table 5-15

STRING-PRECISION GRAPHTEXT SIZE EXAMPLES[a]

Specified Height (TSU)	Resulting Size (Pixels)
1 — 88	7 x 9
89 — 146	14 x 18
147 — 205	18 x 27

[a] These examples assume the default window size is used.

SET GRAPHTEXT SLANT

Specifies the slant each stroke-precision graphtext character has from the vertical.

Host Syntax

```
EcMA  slant-angle
```

Setup Syntax

```
GTSLANT  slant-angle
```

slant-angle: real; specifies the angle each Stroke-precision graphtext character is slanted, in degrees clockwise from the vertical. Must be in the range –32767.0 through 32767.0.

Defaults:	Factory	= 0.0
	Power-Up	= 0.0
	Omitted	= 0.0

If you specify a negative slant, the character is slanted counterclockwise.

SET KEY EXECUTE CHARACTER

Specfies the character that directs the flow of macro expansion.

Host Syntax

```
EcKY  key-execute-char
```

Setup Syntax

```
KEYEXCHAR Sp key-execute-char
```

key-execute-char: integer; specifies the character that causes the terminal to direct the expansion of a macro to the host or the terminal. Valid values are in the range 0 through 127.

Defaults:	Factory	= 16
	Power-Up	= Saved in memory
	Omitted:	= 0

This command sets the value of the key-execute character, used with the DEFINE MACRO command.

Normally, when the operator presses a key which has been programmed (with the DEFINE MACRO command), the characters programmed into the key are sent to the host computer, just as if the operator had typed those characters manually. This includes characters which make up an escape-sequence command for the terminal; the terminal, instead of executing such a command, sends the individual characters to the host.

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, instead, the macro is invoked with an EXPAND MACRO command, then the key-execute character is treated like any other character in the macro definition.

SET LINE INDEX

Specifies the color index for all subsequent lines, panel boundaries, and markers.

Host Syntax

```
EcML line-index
```

Setup Syntax

```
LINEINDEX line-index
```

line-index: integer; indicates the color index for lines, panel boundaries, and markers; must be in the range 0 to 32767 (values greater than 15 set *line-index* to 15).

Defaults:	Factory	= 1
	Power-Up	= 1
	Omitted	= 0

After this command is issued, all new lines, panel boundaries, and markers have the specified color index. This continues until the terminal receives another SET LINE INDEX that changes the color index.

If you specify a line index greater than the maximum index for the particular surface you are drawing on, the terminal uses the maximum index as the line index. The maximum color index for any particular surface is $(2^n)-1$, where n is the number of bit planes assigned to that surface.

A host program assigns a color to a color index with SET SURFACE COLOR MAP. An operator can use the color interface (see *4107/4109 Operators Manual)* or the Setup command CMAP.

SET LINE STYLE

Specifies the line style for subsequent lines and panel boundaries.

Host Syntax

```
EcMV line-style
```

Setup Syntax

```
LINESTYLE line-style
```

line-style: integer; indicates a predefined line style. Must be in the range 0 through 7.

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

Figure 5-13 illustrates the line styles. The terminal will display all new lines, and panel boundaries with the line style specified in this command or with SET 4014 LINE STYLE, whichever was last executed. Existing items are not changed by this command.

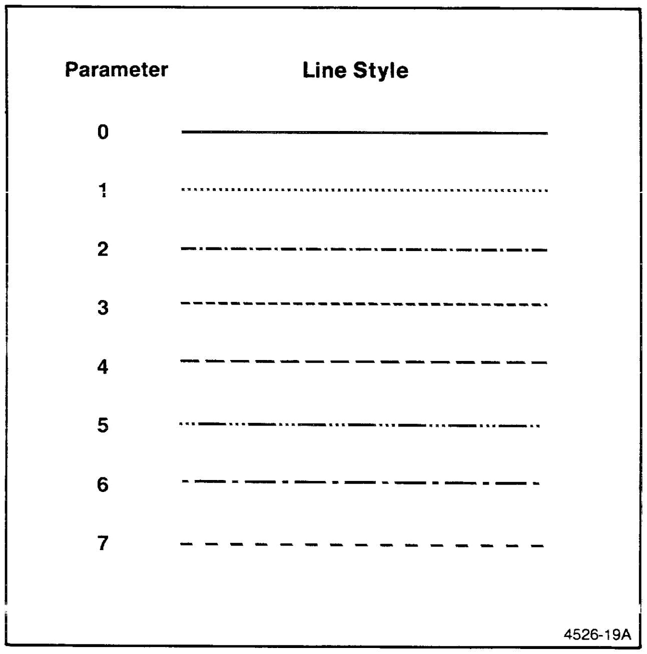


Figure 5-13. Line Styles

SET MARKER TYPE

Specifies the kind of marker to be drawn.

Host Syntax

```
EcMM  marker-number
```

Setup Syntax

```
MARKERTYPE  marker-number
```

marker-number: integer; indicates a predefined marker type. Must be in the range 0 through 10.

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

When you change marker types, markers already displayed remain the same.

Figure 5-14 illustrates the marker types.

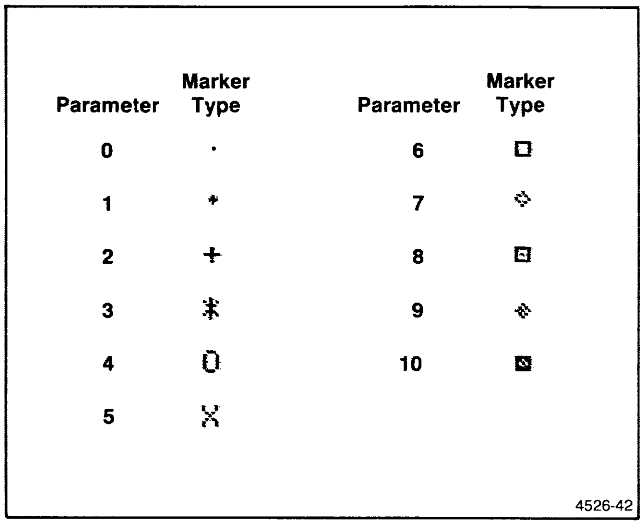


Figure 5-14. Marker Types.

SET PARITY

Specifies the kind of parity the terminal uses when it transmits to the host.

Host Syntax

```
EcNP  parity mode
```

Setup Syntax

```
PARITY  parity mode
```

parity-mode: integer (keyword in Setup syntax); must be one of the following:

Host	Setup	
0	none	Sets parity bit to 0
1	odd	Sets odd parity
2	even	Sets even parity
3	high	Sets parity bit to 1
4	data	Sets no parity; parity bit is used for data

Defaults:	Factory	= 0 (none)
	Power-Up	= Saved in memory
	Omitted	= 0 (none)

This command defines how the parity bit is set on characters output to the host RS-232 port. This setting must match the parity checking scheme that the host uses on incoming data.

With *odd parity* 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. With the *even parity* the eighth bit is set so that the sum of all the bits in the character is even.

The terminal ignores the parity bit in characters it receives from the host.

SET PICK APERTURE

Sets the size of a square aperture in terminal space used to Pick graphics segments.

Host Syntax

```
EcIA aperture-width
```

Setup Syntax

```
GINAPERTURE aperture-width
```

aperture-width: integer; specifies, in terminal space the width of the pick aperture. Valid values range from 0 to 4095.

Defaults:	Factory	= 8
	Power-Up	= 8
	Omitted	= 0

The pick aperture is a square in normalized screen space, centered on the current beam position. During a GIN Pick operation part of the segment Picked must fall within the pick aperture. (For details, see the ENABLE GIN command description, and the description of GIN Pick Reports under the REPORT GIN POINT command description.)

The *aperture-width* parameter sets the width of the pick aperture square, in normalized screen coordinate units. If the width is zero, then part of the segment being Picked must fall exactly at the current cursor position. If the width is 4, then part of the segment being Picked must have its x-coordinate in the range from x0–2 to x0 + 2, and its y-coordinate in the range from y0-2 to y0 + 2, where (x0,y0) is the current beam position in normalize screen space.

The *aperture-width* may be any integer in the range from 0 to 4095. If the parameter is omitted (by terminating the command early), the terminal assumes a value of zero. Very large pick apertures, although allowed, are impractical.

The specified size of the pick aperture is an approximate number. Because of the rounding that occurs when transforming coordinates from terminal space to raster memory space (and vice-versa), it is possible to Pick a line that is slightly outside the stated aperture.

SET PICK APERTURE sets the size of the aperture, or acceptance area around the graphics cursor. The pick aperture is a square in normalized screen space centered on the GIN location. To Pick a segment, the pick aperture must cover at least a portion of the segment you want to Pick. If the width is 4, then part of the segment being Picked must have its x-coordinate in the range from x0 – 2 to x0 + 2, and its y-coordinate in the range from y0 – 2 to y0 + 2, where (x0,y0) is in the GIN cursor position.

SET PICK ID

Marks the next xy location added to the currently open segment as a Pick point and assigns it an identification number.

Host Syntax

```
EcMI  pick-ID-number
```

Setup Syntax

```
SGPICKID  pick-ID-number
```

pick-ID-number: integer; specifies the Pick-ID for parts of the currently open segment. Must be in the range 0 through 32767.

Defaults:		
	Factory	= 1
	Power-Up	= 1
	Omitted	= 0

The Pick-ID numbers are integers in the range from 0 to 32767. Only those parts of a segment which are flagged with positive Pick-ID numbers can be Picked. Pick-ID number 0 marks a part of a segment that cannot be Picked.

The Pick-ID numbers are graphics primitives, but unlike other graphics primitives, they have no direct effect on the appearance of a picture. Like other graphics primitives, Pick-ID numbers are permanent parts of the segment and cannot be changed once the segment has been defined. Therefore, the SET PICK ID command is only valid while a segment is being defined. If you issue a SET PICK ID command while no segment is currently being defined, the terminal detects an error.

At the start of a segment definition, the Pick-ID number is reset to 1. While defining the segment, you can issue SET PICK ID commands to label subsequent parts of the segment with different Pick-ID numbers.

NOTE

To make it easier to create "menus," the INCLUDE COPY OF SEGMENT command does not copy the initial (assumed) Pick-ID.

When a segment is Picked during a graphics input operation, the Pick-ID for the part of the segment within the pick aperture is returned to the host computer. This lets the host computer know not only which segment the operator Picked, but also which part of the segment was Picked.

If, during the Pick operation, the pick aperture is so large as to cover parts of the segment with different Pick-ID numbers, then the terminal returns the last of the Pick-IDs in the display list for parts of the segment within the pick aperture. If, on the other hand, no visible, detectable segment falls within the pick aperture, the terminal sends a GIN Pick Report in which the Pick-ID number is zero.

If several detectable and visible segments are in the Pick aperture, then the highest priority segment is Picked.

SET PIVOT POINT

Specifies the pivot point for subsequent segment definitions.

Host Syntax

```
EcSP  pivot-point
```

Setup Syntax

```
SGPIVOT  pivot-point
```

pivot-point: xy-coordinate; specifies the pivot point for subsequent segment definitions. Values for x and y must be in range 0 through 4095.

Defaults: Factory = (0,0)
Power-Up = (0,0)
Omitted = (0,0)

The pivot point is a point within the segment that defines the position of the segment when that segment is displayed. When a BEGIN SEGMENT command creates a new segment, its pivot point and its position are both set to the xy-coordinate specified by the most recent SET PIVOT POINT command. Once a segment has been defined, its pivot point cannot be changed within the segment definition. However, the position of the whole segment can be relocated within the viewport by using a SET SEGMENT POSITION or a SET SEGMENT IMAGE TRANSFORM command; it is the segment's pivot point that appears at the new position.

The pivot point is the point at which subsequent graphics objects and segments begin. For image transforms (see SET SEGMENT IMAGE TRANSFORM command), the pivot point is the point around which rotation occurs and about which all scaling occurs in both x- and y-directions.

The actual x- and y-coordinate values of the pivot point when the segment is created are only saved in the segment for use by the SAVE and REPORT SEGMENT STATUS commands.

SET PIXEL BEAM POSITION

Sets the position of the pixel beam in the pixel viewport for use by subsequent RASTER WRITE or RUNLENGTH WRITE commands.

Host Syntax

```
EcRH  beam-position
```

Setup Syntax

```
PXPOSITION  beam-position
```

beam-position: xy-coordinate; specifies the position where the next RASTER WRITE or RUNLENGTH WRITE command will take effect. Values for x range from 0 through 639; for y, from 0 through 511.

Defaults: Factory = (0,479)
Power-Up = (0,479)
Omitted = (0,0)

The pixel beam position coordinates are relative to the lower-left corner of the pixel viewport.

If the pixel beam is set to a position outside the pixel viewport, the terminal moves the beam to the nearest pixel inside the viewport. Thus, if the viewport has lower-left and upper-right corners located at (100,100) and (200,200), respectively, a beam position of (150,150) places the beam at the upper-right corner of the viewport.

SET PIXEL VIEWPORT

Sets the pixel viewport position on the pixel writing surface.

Host Syntax

```
EcRS lower-left
     upper-right
```

Setup Syntax

```
PXVIEWPORT lower-left
           upper-right
```

lower-left: xy-coordinate; specifies (in raster memory space) one corner of the pixel viewport. Values for x range from 0 to 639; for y, from 0 to 511.

Defaults:	Factory	= (0,0)
	Power-Up	= Saved in memory
	Omitted	= (0,0)

upper-right: xy-coordinate; specifies the opposite corner of the pixel viewport. Values for x range from 0 to 639; for y, from 0 to 511.

Defaults:	Factory	= (639,479)
	Power-Up	= Saved in memory
	Omitted	= (0,0)

The SET PIXEL VIEWPORT command updates the current pixel beam position to the upper-left corner of the pixel viewport. The *lower-left* and *upper-right* coordinates may actually be the coordinates of any two diagonally opposite corners of the pixel viewport. The terminal will sort the x- and y-coordinate values to determine the correct upper and lower corners.

You can set the viewport's y-direction to include the pixels 480 to 511, which are off the screen. A Level 0 warning will be generated, however.

SET PORT BAUD RATE

Specifies the baud rate for a specified peripheral port.

Host Syntax

```
EcPR port-identifier
     baud-rate
```

Setup Syntax

```
PBAUD port-identifier
      baud-rate
```

port-identifier: string; must be one of the following:

P0:	Specifies Peripheral Port 0
P1:	Specifies Peripheral Port 1

baud-rate: integer; specifies the rate (in bits per second) at which data will be transmitted to the specified port. Valid rates are: 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 4800, and 9600.

Defaults:	Factory	= 2400
	Power-Up	= Saved in memory
	Omitted	= error

This command sets the baud rate for the specified RS-232 peripheral port.

SET PORT EOF STRING

Sets the end-of-file string for the specified peripheral port.

Host Syntax

```
EcPE  port-identifier,
      EOF-string
```

Setup Syntax

```
PEOF  port-identifier,
      EOF-string
```

port-identifier: string; identifies the port to be assigned an end-of-file string. Must be one of the following:

P0: Specifies Peripheral Port 0
P1: Specifies Peripheral Port 1

EOF-string: integer array of up to 10 elements; contains the ADEs of each character in the port's end-of-file string. Each integer in the array must be in the range from 0 through 127. In Setup syntax you can enter ADEs or the ASCII characters themselves.

Defaults:	Factory	= Empty array
	Power-Up	= Saved in memory
	Omitted	= Empty array

This command sets the end-of-file string used when a file is transmitted to the specified RS-232 peripheral port. During a COPY or PLOT to that device, when an end-of-file condition (EOF) is detected, the terminal sends the specified end-of-file string and terminates the PLOT or COPY operation.

SET PORT FLAGGING MODE

Sets the flagging mode for the specified peripheral port.

Host Syntax

```
EcPF  port-identifier
      flagging-mode
      go-character
      stop-character
```

Setup Syntax

```
PFLAG port-identifier
      flagging-mode
      go-character
      stop-character
```

port-identifier: string; specifies the port to be assigned a flagging mode. Must be one of the following:

P0: Specifies Peripheral Port 0
P1: Specifies Peripheral Port 1

flagging-mode: integer (keyword in Setup syntax); must be one of the following:

Host	Setup	
0	none	Specifies no flagging
1	char	Specifies character flagging
2	DTR/CTS	Specifies flagging using the DTR, CTS signal lines

Defaults:	Factory	= 1
	Power-Up	= Saved in memory
	Omitted	= 0

go-character: integer in the range from 0 through 127 (or ASCII character in Setup syntax); specifies the character that indicates the terminal can receive data when character flagging is used. A value of 0 sets $^{D}{}_{1}$ as the go-character.
Defaults: Omitted = 0 (go-character is $^{D}{}_{1}$)

stop-character: small integer in the range from 0 through 127 (or ASCII character in Setup syntax); specifies the character that indicates the terminal is not ready to receive data when character flagging is used. A value of 0 sets $^{D}{}_{3}$ as the stop-character.
Defaults: Omitted = 0 (stop-character is $^{D}{}_{3}$)

This command sets the flagging mode for the specified peripheral port. See the SET FLAGGING MODE command for an explanation of the various flagging modes.

SET PORT PARITY

Specifies the parity scheme for output to the specified peripheral port.

Host Syntax

```
EcPP  port-identifier
      parity-mode
```

Setup Syntax

```
PPARITY  port-identifier
         parity-mode
```

port-identifier: string; identifies the port to be assigned a parity scheme. Must be one of the following:

P0:	Specifies Peripheral Port 0
P1:	Specifies Peripheral Port 1

parity-mode: integer (keyword in Setup syntax); must be one of the following:

Host	Setup	
0	low	Sets parity bit to zero
1	odd	Sets odd parity
2	even	Sets even parity
3	high	Sets parity bit to one
4	none	Specifies no parity; parity bit is omitted on output, and accepted as data on input

Defaults: Factory = 4
Power-Up = Saved in memory
Omitted = 0

The SET PORT PARITY command determines whether characters sent to the specified port have their parity bits set, and if so, how the parity is set.

NOTE

The meanings of the some of the parity-mode parameter settings in this command are NOT the same as those in the SET PARITY command.

Low Parity. If *parity-mode* is set to zero, then low parity is used at the specified port. Each character includes a parity bit. The terminal transmits this bit as a "0" and ignores this bit in characters it receives from this peripheral port.

Odd Parity. If *parity-mode* is 1, then odd parity is used at the specified peripheral port. Each character (or other six-, seven-, or eight-bit data byte) includes a parity bit. In transmitting characters to this port, the terminal sets the parity bit to 1 or 0, whichever is needed to give an odd number of "1" bits (not counting the start and stop bits). The terminal checks for odd parity (that is; for an odd number of "1" bits not counting the start and stop bits) in characters received from this port.

Even Parity. If *parity-mode* is 2, then even parity is used at the specified port. Each transmitted character includes a parity bit. The parity bit is 1 or 0, whichever is necessary to give an even number of "1" bits (not counting the start and stop bits). The terminal checks for even parity in characters received from this port.

High Parity. If *parity-mode* is 3, then high parity is used at the specified port. Each character includes a parity bit. The terminal always transmits this bit as a "1". The terminal ignores the parity bit in characters received from this port.

None Parity. If this parameter is 4, the parity bit is omitted on output and accepted as data on input.

For normal operation with the ASCII character set, there should be seven data bits, one parity bit, and one or two stop bits, depending on your host. Therefore, the SET PORT PARITY command should have a parity setting of 0, 1, 2, or 3.

SET PORT STOP BITS

Sets the number of stop bits and the number of data bits in characters sent to the specified port.

Host Syntax

```
EcPB port-identifier
     number-of-stop-bits
     number-of-data-bits
```

Setup Syntax

```
PBITS port-identifier
      number-of-stop-bits
      number-of-data-bits
```

port-identifier: string; must be one of the following:

P0:	Specifies Peripheral Port 0
P1:	Specifies Peripheral Port 1

number-of-stop-bits: integer; specifies the number of stop bits in characters sent to the specified port. Must be 1 or 2.

Defaults:	Factory	= 2
	Power-Up	= Saved in memory
	Omitted	= Error

number-of-data-bits: integer; specifies the number of data bits in characters sent to the specified port. Must be 5, 6, 7 or 8. This count does not include the parity bit, whose presence or absence is determined by the SET PORT PARITY command.

Defaults:	Factory	= 8
	Power-Up	= Saved in memory
	Omitted	= Error

This command sets the number of stop bits and data bits in characters sent between the terminal and the specified peripheral port.

Each character has a start bit, which is always 0. After the start bit come from five to eight data bits; the number of data bits is determined by the SET PORT STOP BITS command. After the data bits comes the parity bit; this bit may be omitted, depending on the most recent SET PORT PARITY command. After the parity bit come one or two stop bits. The number of stop bits is determined by the most recent SET PORT STOP BITS command. The stop bits are always 1.

SET PROMPT STRING

Specifies the string that initiates the terminal's Prompt mode.

Host Syntax

```
EcNS prompt-string
```

Setup Syntax

```
PROMPTSTRING prompt-string
```

prompt-string: integer array of small integers (delimited character string in Setup syntax). Each integer must be in the range from 0 through 127; each is the ADE of an ASCII character. The maximum length of the string defined by this parameter is 10 characters.

Defaults:	Factory	= Empty array
	Power-Up	= Saved in memory
	Omitted	= Empty array

This command defines the string that the terminal recognizes as a prompt string from the host.

In Setup mode you can enter either a delimited string of ASCII characters or a series of ADEs. ADEs less than ten must be preceded by a zero. For example, to specify a Back Space character, use the ADE **08**; the character 8 would be interpreted as the numeral 8 with ADE 56.

See the *Communications* section for an explanation of Prompt mode.

SET QUEUE SIZE

Specifies the size (in bytes) of the terminal's input queue.

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
Power-Up = Saved in memory
Omitted = Error

This command reserves part of the terminal's 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 input queue fills up and the terminal is not using a flagging mechanism, the terminal simply discards incoming data until there is more room in the queue. See the *Communications* section for details.

Specifying a very large input queue may affect the terminal's ability to store and display graphics information. Specifying a very small input queue may cause data to be lost when the input queue overflows.

SET REPORT EOM FREQUENCY

Specifies how often the terminal sends EOM indicators to the host.

Host Syntax

```
EcIM EOM-frequency
```

Setup Syntax

```
REOM EOM-frequency
```

EOM-frequency: integer; specifies whether EOM indicators should be sent more or less frequently in reports to the host. "More frequently" means at the end of each part of the message; "less frequently" means only when needed to prevent the maximum line length from being exceeded. Valid values are:

Host	Setup	
0	0	Less frequently
1	1	More frequently

Defaults: Factory = 1
Power-Up = 1
Omitted = 1

This command controls how frequently the terminal intersperses EOM indicators among the characters that comprise a "report message" that the terminal sends to the host computer.

Generally speaking, setting the *EOM-frequency* to 1 (more frequent) causes a long message to be broken into separate lines of text for each part of the report. For instance, each GIN REPORT within a GIN REPORT SEQUENCE would occupy a separate line of text, terminated with an EOM indicator.

Setting the *EOM-frequency* to 0 (less frequent) permits several parts of a report to fit on the same line of text. For instance, several GIN Reports could fit on the same line. The line is terminated with an EOM indicator only when the terminal's maximum line length is about to be exceeded.

EOM Indicators. The EOM indicator (end-of-message indicator) serves to mark the end of a "line of text" in data being sent to the host.

The EOM indicator is always the current end-of-line string, as defined by the most recent SET EOL STRING command.

SET REPORT MAX LINE LENGTH

Specifies the maximum number of characters, within a line, the terminal sends to the host.

Host Syntax

```
EcIL  max-line-length
```

Setup Syntax

```
RLINELENGTH  max-line-length
```

max-line-length: integer; specifies the maximum number of characters per line in reports sent to the host. Setting this parameter to zero disables the maximum-line-length feature. Valid values range from 0 through 65535.

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

This command sets the maximum line length for report messages which the terminal sends to the host computer. It also determines how frequently the terminal intersperses EOM indicators among data being transferred to the host in a PLOT command.

EOM Indicators. The terminal uses EOM indicators (end-of-message indicators) to force the end of a line of text in data it sends to the host. The terminal's EOM indicator is always the same as its EOL string, set by the SET EOL STRING command.

Report Messages. If the terminal has a report to send to the host, and that report would cause the maximum line length to be exceeded, then the terminal inserts an EOM indicator into the report. The EOM indicator serves to terminate the current line of text.

The exact places where EOM indicators may be inserted are described in the syntax of the particular report type. For details, see the description of the individual reports.

Other Data Sent to the Host. When the terminal sends data to the host in response to a plot command, it intersperses EOM indicators in that data, so as to break the data into "lines of text." Each such line of text has the maximum line length, as determined by the SET REPORT MAX LINE LENGTH command.

This feature is to accommodate host computers that cannot reliably accept lines of more than a certain number of characters. The feature can be disabled by setting the terminal's maximum line length to zero.

NOTE

Even if the data being transferred already contains CR characters (or other EOM characters), the terminal still inserts a CR (or other EOL string) after every max-line-length characters of data.

Therefore, if you will be transferring data that is already broken into individual lines of text, you should set the terminal's max report line length to zero, thereby disabling the "maximum line length" feature. (Alternatively, you can set the EOL string to the empty string.)

SET REPORT SIG CHARACTERS

Assigns signature characters used within report messages that the terminal sends to the host.

Host Syntax

```
EcIS report-type-code
     signature-character
     terminal-signature-character
```

Setup Syntax

```
RSIGCHARS report-type-code
          signature-character
          terminal-signature-character
```

report-type-code: integer; -3 to -1, or a valid GIN device function. If this is a GIN device-function code, then the signature characters defined in this command will be used in GIN REPORT SEQUENCES for that device-function code. If this parameter is -3, then these signature characters are used in non-GIN report messages. If the parameter is -2, these signature characters are used only when responding to REPORT GIN POINT: -2 commands. If the parameter is -1, then these signature characters are used in all report messages (both GIN reports and non-GIN reports).
Defaults: Omitted = 0

signature-character: integer; the numeric equivalent of the ASCII character which is to be used as the *signature-character* in reports of the specified type. If the *signature-character* is set to NU — numeric equivalent of zero — then that *signature-character* is omitted from reports sent to the host.

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

terminal-signature-character: integer; specifies the character to be used as the *terminal-signature-character* in reports of the specified type. If the *terminal-signature-character* is NU, then it is omitted in reports sent to the host.

Defaults:	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

This command assigns the signature characters to be used within report messages that the terminal sends to the host computer.

The number -1 specifies "all reports" — inquiry reports and reports for all GIN device-function combinations.

NOTE

If GIN is enabled for more than one device at a time, then, different signature-characters and terminal-signature-characters are required for each enabled GIN device. This is necessary in order that the host computer can parse the interleaved GIN Report Sequences occurring when more than one GIN device is active.

Also, if non-GIN reports are requested while GIN is enabled, then different signature-characters and terminal-signature-characters are needed to distinguish non-GIN reports from GIN reports.

On power-up, all signature characters are initialized to NU.

SET SEGMENT CLASS

Alters the classes that a segment is assigned to for use in segment class matching operations.

Host Syntax

```
EcSA segment-number
     removal-array
     addition-array
```

Setup Syntax

```
SGCLASS segment-number
        removal-array
        addition-array
```

segment-number: integer; names the segment whose classes are being altered. Valid values for *segment-number* are:

−3	All segments that match the current matching class
−2	The default for segments not yet defined
−1	All segments
1 — 32767	A specific segment

Defaults: Omitted = Error

removal-array: integer array; lists the segment classes from which the specified segment is removed. Valid values for inclusion in *removal-array* are:

−1	All classes
1 — 64	A specific class

Defaults: Factory = Empty array
Power-Up = Empty array
Omitted = Empty array

addition-array: integer array; lists the segment classes to which the specified segment is added. Valid values for inclusion in *addition-array* are:

−1	All classes
1 — 64	A specific class

Defaults: Factory = Empty array
Power-Up = Empty array
Omitted = Empty array

Each segment belongs to a set of classes. This command alters this set by first removing the segment from the classes specified in *removal-array* and then adding the segment to the classes specified in *addition-array*. Removing a class that is not in a segment's set, or adding a class that is already there, does not cause an error.

SET SEGMENT DETECTABILITY

Sets the detectability of a segment.

Host Syntax

```
EcSD segment-number
     detectability
```

Setup Syntax

```
SGDETECT segment-number
         detectability
```

segment-number: integer; names the segment whose detectability is being set. Valid values for *segment-number* are:

−3	All segments that match the current matching class
−2	The default for segments not yet defined
−1	All segments
1 — 32767	A specific segment

Defaults: Omitted = Error

detectability: integer (keyword in Setup syntax); specifies whether a segment can be Picked in a GIN Pick operation or not. Must be one of the following:

Host	Setup	
0	no	Can't be Picked
1	yes	Can be Picked

Defaults: Factory = 1
Power-Up = 1
Omitted = 0

If a *detectability* of 1 is specified, and the segment is visible in the current view, it is Picked when the Pick function is enabled. If 0 is specified, the segment is not Picked.

If graphics input (GIN) is in progress and the segment specified is being used as the GIN cursor, this command has no effect. A segment is not Picked when it is being actively used as a GIN cursor.

SET SEGMENT DISPLAY PRIORITY

Sets the display priority of a specified segment.

Host Syntax

```
EcSS segment-number
     priority-number
```

Setup Syntax

```
SGPRIORITY segment-number
           priority-number
```

segment-number integer; names the segment for which a display priority is being set. Valid values for *segment-number* are:

–3	All segments that match the current matching class
–2	The default for segments not yet defined
–1	All segments
1 — 32767	A specific segment

Defaults: Omitted = Error

priority-number: integer; specifies the display priority of the specified segment. Must be in the range –32768 through 32767.

Defaults: Factory = 0
Power-Up = 0
Omitted = 0

NOTE

During a graphics input Pick operation, the display priority of segments affects which of several eligible segments will be Picked. (An eligible segment is a visible, detectable segment, part of which falls within the current pick aperture.) The eligible segment with the highest priority is the one that will be Picked.

On power-up, the default priority for new segments is 0, unless the display priority for segment –2 is set to a different number by SET SEGMENT DISPLAY PRIORITY command.

Upon redisplay (when an invisible segment is made visible, or when a view is renewed), the segments are displayed in priority order, so that the segment(s) last displayed are those with the highest display priority. For instance, if Segment 56 has a priority number of 1, and Segment 38 has priority number of 2, Segment 38 is displayed last and appears to be in front of Segment 56.

For segments that are assigned the same display priority number, the order of display and the order of Picking are not defined, and may be different on different Tektronix terminals.

SET SEGMENT HIGHLIGHTING

Turns highlighting on or off for specified segments.

Host Syntax

```
EcSH segment-number
     highlighting
```

Setup Syntax

```
SGHIGHLIGHT segment-number
            highlighting
```

segment-number: names the segment for which highlighting is being specified. Valid values for *segment-number* are:

−3	All segments that match the current matching class
−2	The default for segments not yet defined
−1	All segments
1 — 32767	A specific segment

Defaults: Omitted = Error

highlighting: integer (keyword in Setup syntax); specifies whether a segment is highlighted (blinks) or not.

Host	Setup	
0	no	Turns off the blink feature
1	yes	Turns on the blink feature

Defaults: Factory = 0
Power-Up = 0
Omitted = 0

Highlighting a segment makes it blink by turning the illuminated pixels off and on. Segment highlighting is performed regardless of the current fixup level. An open segment does not blink, but will start blinking when it is closed.

NOTE

If there are many segments, or complex segments with many pixels, the highlighting process may be very slow. In that case, you may find it advantageous not to use the SET SEGMENT HIGHLIGHTING command. Instead, you can place the segments to be highlighted in a different view with a viewport on another surface, and blink that surface with the SET SURFACE VISIBILITY command with a parameter value of 2.

SET SEGMENT IMAGE TRANSFORM

Scales, rotates, and positions a specified segment.

Host Syntax

```
EcSI segment-number
     x-scale-factor
     y-scale-factor
     rotation-angle
     position
```

Setup Syntax

```
SGTRANSFORM segment-number
            x-scale-factor
            y-scale-factor
            rotation-angle
            position
```

segment-number: integer; names the segment for which an image transform is being specified. Valid values for *segment-number* are:

−3	All segments that match the current matching class
−2	The default for segments not yet defined
−1	All segments
1 — 32767	A specific segment

Defaults: Omitted = Error

x-scale-factor: real; specifies the factor by which the segment is scaled in the x-direction. Must be in the range −32767 through 32767.

Defaults: Factory = 1.0
Power-Up = 1.0
Omitted = 0.0

y-scale-factor: real; specifies the factor by which the segment is scaled in the y-direction. Must be in the range −32767 through 32767.

Defaults: Factory = 1.0
Power-Up = 1.0
Omitted = 0.0

rotation-angle: real; specifies the counterclockwise rotation angle, in degrees. (A negative number specifies a clockwise rotation.) Must be in the range −32767 through 32767.

Defaults: Factory = 0.0
Power-Up = 0.0
Omitted = 0.0

position: xy-coordinates; specify the position in terminal space where the segment's pivot point is to be located. Both x and y must be in the range 0 through 4095.

Defaults:	Factory	= (0,0)
	Power-Up	= (0,0)
	Omitted	= (0,0)

The SET SEGMENT IMAGE TRANSFORM command transforms the segment as follows:

1. First, the segment is scaled in the x- and y-directions by the amounts specified by the x- and y-scale factors. A negative scale factor indicates an inversion about the appropriate axis.

2. Next, the segment is rotated counterclockwise about its pivot point by the number of degrees specified in the rotation parameter. If this parameter is negative, then the segment is rotated clockwise.

3. Finally, the segment is moved (translated) so that its pivot point is at the position specified by the position xy parameter.

Image transform operations are not cumulative. They always start at the size and position of the original segment definition.

The current fixup level determines how soon the display is updated to show the new position of the transformed segment (see SET FIXUP LEVEL for details).

Specifying 0 for *segment-number* is not allowed. (Segment 0 refers to the crosshair cursor, which cannot be rotated or scaled. If you want to change the position of Segment 0, use the SET SEGMENT POSITION command.)

Specifying –2 for *segment-number* allows you to set the image transform parameters for segments which have not yet been defined. However, if the x- and y-scale factors are not unity, if the rotation angle is not zero, or if the position parameter is not the same as the current pivot point, then a segment being defined will not be displayed until an END SEGMENT command terminates the segment definition.

After the scaling, rotation, and positioning has occurred, any parts of the segment that extend outside the current window are not displayed.

NOTE

When using SET SEGMENT IMAGE TRANSFORM or SET SEGMENT POSITION, avoid positions that extend segment parts to x or y coordinates greater than 8091 or less than -4096. Segments extending that far outside the normal 0 to 4095 terminal space may not be displayed properly.

If a segment is being used as the graphics cursor for a GIN operation, moving that segment with the SET SEGMENT IMAGE TRANSFORM command does not change the current graphics input location. The next time the operator moves the joydisk (or other GIN device), the graphics input location is changed and the segment's pivot point is moved to that new graphics input location.

If a segment's position is changed by the SET SEGMENT POSITION command, or by using the segment as a GIN cursor, then the current image transform for that segment is updated to reflect the change.

SET SEGMENT POSITION

Moves the pivot point of a segment to a specified position in terminal space.

Host Syntax

```
EcSX segment-number
     position
```

Setup Syntax

```
SGPOSITION segment-number
           position
```

segment-number: integer; names the segment whose position is being specified. Valid values for *segment-number* are:

-3	All segments that match the current matching class
-2	The default for segments not yet defined
-1	All segments
0	The crosshair cursor
1 — 32767	A specific segment

Defaults: Omitted = 0

position: xy-coordinate; specifies the position in terminal space of the segment's pivot point. Both x and y must be in the range 0 through 4095.

Defaults: Factory = (0,0)
Power-Up = (0,0)
Omitted = (0,0)

A segment number in the range from 1 to 32767 specifies a segment which has previously been defined with BEGIN SEGMENT and END SEGMENT commands. An error is detected if the specified segment does not exist.

Normally, when a segment is defined, its position is set to the same location as its pivot point (set by the most recent SET PIVOT POINT command). However, you can issue a SET SEGMENT POSITION command in which *segment-number* is -2 and *position* specifies a different location from the current pivot point.

Issuing a SET PIVOT POINT command cancels the effect of any previous SET SEGMENT POSITION commands for Segment -2.

If part of a segment extends outside the current window, then that part of the segment is not displayed.

A segment's position may also be changed by the SET SEGMENT IMAGE TRANSFORM command, or by using the segment as the graphics input cursor.

If the segment specified is being used as the graphics cursor in a GIN operation, then the segment's position is changed by the SET SEGMENT POSITION command, but the current graphics input location is not updated. The graphics input location will be updated, and the segment serving as the graphics cursor will be moved, the next time the operator moves the joydisk (or other graphics input device).

SET SEGMENT VISIBILITY

Sets the visibility attribute of a specified segment(s).

Host Syntax

```
EcSV segment-number
     visibility
```

Setup Syntax

```
SGVISIBILITY segment-number
             visibility
```

segment-number: integer; names the segment whose visibility is being specified. Valid values for *segment-number* are:

-3	All segments that match the current matching class
-2	The default for segments not yet defined
-1	All segments
0	The crosshair cursor
1 — 32767	A specific segment

Defaults: Omitted = 0

visibility: integer(keyword in Setup syntax); specifies whether a segment is visible in the current view or not. Must be one of the following:

Host	Setup	
0	no	Makes segment invisible
1	yes	Makes segment visible

Defaults: Factory = 1
Power-Up = 1
Omitted = 0

If a *visibility* of 0 (invisible) is specified for a segment in the current view, the segment is made invisible either immediately or the next time the screen is erased, depending on the fixup level specified in the most recent SET FIXUP LEVEL command.

If a *visibility* of 1 (visible) is specified, the segment appears in the mode specified by the most recent SET SEGMENT WRITING MODE command for that segment.

Segments are automatically visible only in the view in which they are created; however, segments exist in all views, but must specifically be made visible in any view other than the one in which they are created.

"The view in which a segment is created" means the currently selected view at the time the END SEGMENT command terminates the segment definition.

The default for new segments is visible. This may, however, be altered by a SET SEGMENT VISIBILITY command for Segment -2.

NOTE

When GIN is enabled, the terminal saves the visibility attribute for the segment which is to be used as the graphics cursor. That segment is then made visible. When GIN is disabled, the saved value of the visibility attribute is restored to the segment which acted as the graphics cursor.

Therefore, any SET SEGMENT VISIBILITY commands, which may be issued for the graphic cursor while GIN is enabled, will cease to have effect when GIN is disabled.

SET SEGMENT WRITING MODE

Selects the writing mode used when displaying a specified segment.

Host Syntax

```
EcSM segment-number
     writing-mode
```

Setup Syntax

```
SGMODE segment-number
       writing-mode
```

segment-number: integer; names the segment for which a writing mode is being specified. Valid values for *segment-number* are:

–3	All segments that match the current matching class
–2	The default for segments not yet defined
–1	All segments
0	The GIN cursor
1 — 32767	A specific segment

Defaults: Omitted = 0

writing-mode: integer (keyword in Setup syntax); selects which writing mode is used. Must be one of the following:

Host	Setup	
1	set	Selects Set mode
2	XOR	Selects XOR mode

Defaults: Factory = 1
Power-Up = 1
Omitted = Error

When Set mode is selected to write an image of the segment into the current viewport, each pixel being written over is set to the appropriate color index. The color index is the current line index for pixels which form the image of lines or graphtext, and the current text index for pixels which make up characters of alphatext or graphtext. The previous color index stored in the pixel is destroyed.

When XOR (exclusive OR) mode is selected, each pixel being written over is replaced by a pixel in a new color index. The new color index is a binary number which is the bit-by-bit "exclusive OR" of the bits in the old color index for that pixel and the corresponding bits of the current line index or text index.

The XOR mode is convenient for writing images which may later need to be erased or re-positioned on the screen. This is because a line (or alphatext character, etc.) can be erased by writing over it again in XOR mode. (This property is a consequence of the Boolean logic theorem that (A XOR B) XOR B = A.)

If the segment specified is currently being used as a graphic cursor, then the SET SEGMENT WRITING MODE command does not take effect until the current GIN (graphics input) operation is disabled.

NOTE

It's a good idea to issue a PAGE or RENEW VIEW command after changing a segment's writing mode. This ensures that the viewport is updated to display the segment properly.

If you don't do this, and the current fixup level is five or more, it is possible that repositioning a segment may cause it to appear in more than one location on the screen. The remedy is for the host to issue a PAGE or RENEW VIEW command, or for the operator to press the G Eras key.

SET SNOOPY MODE

Specifies whether or not the terminal is in Snoopy mode.

Host Syntax

```
EcKS snoopy-mode
```

Setup Syntax

```
SNOOPY snoopy-mode
```

snoopy-mode: integer (keyword in Setup syntax); 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 (no)
Power-Up = 0 (no)
Omitted = 1 (yes)

When the terminal is in Snoopy mode, control characters that are normally used to control the terminal are displayed instead of executed (except for LF, which is displayed and causes a new display line). Characters that are normally filtered out of the host's data stream (such as prompt sequences) are still filtered and not displayed. You cannot execute commands from the host or keyboard when the terminal is in Snoopy mode.

Snoopy mode cannot be terminated from the host; only the operator can do so. To terminate Snoopy mode, press the Cancel key or enter **SNOOPY no** in Setup mode.

SET STOP BITS

Specifies number of stop bits appended to each character the terminal transmits.

Host Syntax

```
EcNB number-of-stopbits
```

Setup Syntax

```
STOPBITS number-of-stopbits
```

number-of-stopbits: integer; indicates the number of stop bits appended to each character the terminal transmits; must be 1 or 2.

Defaults: Factory = 1
Power-Up = Saved in memory
Omitted = error

See the *Communications* section for an explanation of stopbits.

SET SURFACE COLOR MAP

Sets the color map for the graphics region.

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. –1 means a "super surface" consisting of all bit planes of all surfaces presently defined.

color-mixtures: integer array; assigns a color mixture to one or more color indices. Each array is a quadruple (group of four). The first integer in each quadruple names a color index, while the following three integers specify the color mixture for that color index. The color mixture is specified in the HLS, RGB, or CMY coordinate system, according to the most recent SET COLOR MODE command. A blinking color can be specified by adding 1000 to the third coordinate of an index. Table 5-16 (under SET SURFACE DEFINITIONS) lists the default color mixtures.

Index 0 is the background color, which is behind all the writing surfaces.

The valid ranges for the first, second, and third coordinates in each system are, respectively:

HLS:	-32768 to 32767
	0 to 100
	0 to 100, or 1000 to 1100
RGB and CMY:	0 to 100
	0 to 100
	0 to 100, or 1000 to 1100

The SET SURFACE COLOR MAP command sets the "color map" for the graphics region (the bit planes). It determines the mapping from that surface's color indices to particular color mixtures. It can be used to set the background color by setting the color index to 0.

Use the SET DIALOG AREA COLOR MAP command to set the dialog area color map.

The effect of the SET SURFACE COLOR MAP command continues until superseded by another SET SURFACE COLOR MAP command, or until surfaces are redefined with a SET SURFACE DEFINITIONS command.

NOTE

Unlike the 4105, color maps for the 4107 and 4109 are not saved in nonvolatile memory.

In this command, Surface –1 means a "super-surface" consisting of all bit planes in all surfaces presently defined. This is for use in advanced applications, such as controlling the exact color displayed where images on one surface overlap images on another surface.

Adding 1000 to the third color coordinate of the index causes the color to blink by alternately becoming visible and invisible. For example, in the HLS color mode a normal red is indicated by (120,50,100), and a blinking red is given by (120,50,1100).

SET SURFACE DEFINITIONS

Erases the screen and sets the number of surfaces and the number of bit planes in each surface.

Host Syntax

```
EcRD surface-defs
```

Setup Syntax

```
SDEFINITIONS surface-defs
```

surface-defs: integer array; specifies the number of bit planes for each surface.

Defaults: Factory = 4 (Surface 1 with four bit planes)
Power-Up = 4 (Surface 1 with four bit planes)
Omitted = Error RD11

The SET SURFACE DEFINITIONS command erases the screen and sets the number of bit planes in each surface. Initially, all planes are allocated to one surface. Normal priority is front to back, with surface 1 being the front.

The *number-of-bit-planes* in each surface determines the maximum color index which may be written into pixels on that surface. A surface with *n* bit planes is allowed color indices from 1 to 2^n-1.

If more planes are specified than the terminal has, the excess specifications are ignored.

Initializing Color Mixtures. The SET SURFACE DEFINITIONS command assigns a color mixture to each possible color index on each surface being defined. Table 5-16 lists the default mixtures.

Table 5-16

DEFAULT COLOR MIXTURES FOR SURFACES

Color Index	Color Mixture	Color Coordinates		
		H	L	S
0	Transparent	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
8	Red-Yellow (Orange)	150	50	100
9	Green-Yellow	210	50	100
10	Green-Cyan	270	50	100
11	Blue-Cyan	330	50	100
12	Blue-Magenta	90	50	100
13	Red-Magenta	90	50	100
14	Dark-Gray	0	33	0
15	Light-Gray	0	66	0

Initializing Visibility. The SET SURFACE DEFINITIONS command causes each surface it defines to be visible, as if a SET SURFACE VISIBILITY command were issued for that surface.

Initializing Surface Priorities. The SET SURFACE DEFINITIONS command arranges the surfaces in numerical order, from front to back. Surface 1 is in front; Surface 4 (if it exists) is in back. The background, whose color is set by SET BACKGROUND COLOR, is always behind all of the defined writing surfaces. You can change this ordering with the SET SURFACE PRIORITIES command.

SET SURFACE PRIORITIES

Sets the priority of each writing surface.

Host Syntax

```
EcRN priorities
```

Setup Syntax

```
SPRIORITIES priorities
```

priorities: integer array; grouped in pairs. The first integer in each pair is a surface number, and the second integer of the pair is a priority number for that surface. Each surface is "in front of" all surfaces that have a larger priority number than it, and "behind" all surfaces that have lesser priority numbers. Valid surface and priority values range from 1 through 4.

Defaults: Factory = 1,1
Power-Up = 1,1
Omitted = Error RN11

This command lists the surfaces, front to back; it determines which of the transparent writing surfaces are "in front" of others, and which are "behind" other surfaces.

Objects (especially filled areas) drawn on one surface will obscure objects drawn on a surface that is "behind" that surface (see the SET COLOR MODE command description).

Special Cases. If two surfaces are assigned the same priority, then the lower-numbered surface is deemed to be in front of the higher-numbered surface. If more than one priority is assigned to the same surface, then the last priority assigned that surface is the one which has effect.

Default. On power-up, only Surface 1 exists. When multiple surfaces are defined (with the SET SURFACE DEFINITIONS command), the surface priorities are: Surface 1 is the front surface; Surface 2 is behind Surface 1; Surface 3 is behind Surface 2; and Surface 4 is behind Surface 3.

SET SURFACE VISIBILITY

Sets the visibility of one or more surfaces without affecting surface priorities.

Host Syntax

```
EcRI surface-numbers-and-visibilities
```

Setup Syntax

```
SVISIBILITY surface-numbers-and-visibilities
```

surface-numbers-and-visibilities: integer array; holds pairs of integers. The first integer in each pair is a surface number. This must be in the range from 1 to 4. The second integer in each pair specifies the visibility mode, and must be in the range from 0 to 2:

0	Invisible (no objects displayed)
1	Visible
2	Blinking (alternates between visible and invisible)

Defaults: Factory = 1,1
Power-Up = 1,1
Omitted = Error RI11

This command changes the visibility of one or more surfaces without affecting surface priorities or the visibility attributes of any segments in the surfaces. Segments on an invisible surface cannot be Picked during a GIN operation. Visibility mode 2 causes a surface to *blink,* or alternate between being visible and being invisible at a rate of (0.75 sec on, 0.75 sec off). Blinking indices and blinking surfaces blink in unison; highlighted segments do not necessarily blink in phase with each other or with other blinking graphics objects.

SET TAB STOPS

Sets tab stop(s) at the specified position(s).

Host Syntax

```
EcKB tab-positions
```

Setup Syntax

```
TABS tab-positions
```

tab-positions integer array (can also be a keyword in Setup mode) specifying one or more tab stops. The keyword **all** can be used in Setup mode to set tab stops at every column. Specify 0 to clear all tab stops; specify −1 or **all** to set tab stops at every column; specify −2 to reset tab stops to factory default.

Defaults:	Factory	= 1, 9, 17, 25, 33, 41, 49, 57, 65, 73, 80
	Power-up	= Saved in memory
	Omitted	= 0

SET TEXT INDEX

Specifies the color index for alphatext and graphtext in the graphics area.

Host Syntax

```
EcMT text-index
```

Setup Syntax

```
GTINDEX text-index
```

text-index: integer; indicates color index for text in the graphics area. Must be in the range 0 through 65535 (values greater than 15 set *text-index* to 15).

Defaults:	Factory	= 1
	Power-Up	= 1
	Omitted	= 0

This command sets the color index for all new text displayed in the graphics region. This includes all graphtext and alphatext displayed when the dialog area is disabled. This command does not change the color index of existing text.

Alphatext displayed in the dialog area is not affected by this command. Use SET DIALOG AREA INDEX to set the color index of dialog area alphatext.

If text is displayed on a surface that has fewer than 4 bit planes, and if you specify a text index greater than the maximum index for that surface, the terminal uses the maximum index as the text index. The maximum color index for any particular surface is 2^n-1, where n is the number of bit planes assigned to that surface.

SET TABLET HEADER CHARACTERS

Selects the key characters used in GIN Stroke Reports.

Host Syntax

```
-----------------------------------------------------------------
| EcIH character-set-selector                                    |
-----------------------------------------------------------------
```

Setup Syntax

```
-----------------------------------------------------------------
| GINSHEADERCHARS character-set-selector                         |
-----------------------------------------------------------------
```

character-set-selector: integer (keyword in Setup syntax); must be one of the following:

Host	Setup	
0	letters	Selects Z, J, and 0 as key characters
1	control	Selects Z, S_B , and U_S

Defaults: Factory = 0
Power-Up = 0
Omitted = 0

If character-set-selector is 0 (**letters** in Setup), the key characters in GIN Stroke Reports are:

- o Z, 1, 2, or 3 for the first point in a Stroke
- o J for subsequent points in a Stroke
- o 0 for the last point in a Stroke

If character-set-selector is 1 (**control** in Setup), the key characters in GIN Stroke Reports are:

- o Z, 1, 2, or 3 for the first point in a Stroke
- o S_B for subsequent points in a Stroke
- o U_S for the last points in a Stroke

For more details, see the GIN Stroke Report description under REPORT GIN POINT.

SET TRANSMIT DELAY

Specifies the terminal's transmit delay between transmitting lines of text.

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
	Power-Up	= Saved in memory
	Omitted	= 0

After the terminal sends an end-of-message character, it 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. The actual delay time may be up to 33 milliseconds longer than the time specified by this command, because of the resolution of the terminal's internal timer.

If the terminal is in Prompt mode, the terminal waits both until the transmit delay has elapsed *and* until it receives a prompt sequence before transmitting another line of text.

SET TRANSMIT RATE LIMIT

Specifies the effective transmit baud rate limit.

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
	Power-Up	= Saved in memory
	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 so that it does not exceed the indicated rate limit.

SET VIEW ATTRIBUTES

Sets the surface, wipe index, and border index for the current view.

Host Syntax

```
EcRA surface-number
     wipe-index
     border-index
```

Setup Syntax

```
VIEWATTRIBUTES surface-number
               wipe-index
               border-index
```

surface-number: integer; specifies the surface on which the view's viewport is located. Valid values are −1 through 4.

Defaults:		
	Factory	= 1
	Power-Up	= 1
	Omitted	= 0

wipe-index: integer; the color index used for wiping (erasing) the viewport. Must be in the range 0 through 65535.

Defaults:		
	Factory	= 0
	Power-Up	= 0
	Omitted	= 0

border-index: integer; the color index used for displaying a border around the viewport. Must be in the range 0 through 65535.

Defaults:		
	Factory	= 1
	Power-Up	= 1
	Omitted	= 0

The view most recently selected with the SELECT VIEW command or the View key is the current view.

Surface Number. If Surface 0 is specified, the current surface for the view is left unchanged. If Surface −1 is specified, then a super surface is used. This super surface consists of all bit planes on all of the presently defined surfaces. If a surface is specified that does not exist, then an error occurs. (A surface does not exist if it was not defined in the most recent SET SURFACE DEFINITIONS command.)

Wipe Index. If you specify a wipe index greater than the maximum index for the surface specified, the terminal uses the maximum index as the wipe index. The maximum color index for any particular surface is 2^n-1, where n is the number of bit planes assigned to that surface.

Border Index. If a border index greater than the maximum index for the surface is specified, the terminal uses the maximum index as the border index. (The border is drawn only if the BORDER key or SET BORDER VISIBILITY command has made the border visible.)

SET VIEW DISPLAY CLUSTER

Specifies a list of views that are to have identical windows.

Host Syntax

```
EcRQ view-numbers
```

Setup Syntax

```
VDISPLAYCLUSTER view-numbers
```

view-numbers: integer array; specifies which of the terminal's views are to have identical windows. Must be –2 through 64:

–2	Removes all views in the current view's cluster.
–1	Clusters all 64 possible views together in one display cluster. (Any views that might later be created — with the SELECT VIEW command — will also be included in this display cluster.)
0	Refers to the current view, as selected by the most recent SELECT VIEW command.
1 — 64	Names a specific view; a view need not exist to be included in a view cluster. (Later, when the view is created with the SELECT VIEW command, it will be included in the view cluster.)

Defaults: Omitted (Setup only) = Remove all views from all clusters

In some circumstances, you may want to have several views with identical windows for their window-viewport transforms. The SET VIEW DISPLAY CLUSTER command lets you specify a list of views that are to have identical windows. Changing the window for any view in this cluster also changes the window for every other view in the cluster. Also, renewing any view in this cluster (RENEW VIEW or PAGE command, or G Eras or S Eras key) also renews every other view in the cluster.

Removing Specific Views From a Display Cluster. A view cannot belong to more than one display cluster. Thus, including a view in one cluster automatically removes it from any other clusters.

Removing All Views From All Clusters. All views are removed from all clusters if the SET VIEW DISPLAY CLUSTER command's integer array parameter is empty (if, in Host syntax, the array count is 0, or in Setup syntax, no view numbers are listed).

Removing All Views From Only One Display Cluster. To remove all views from a single cluster:

1. Issue a SELECT VIEW command to select one of the views in the cluster.

2. Issue a SET VIEW DISPLAY CLUSTER command containing the special view number, –2.

SET VIEWPORT

Sets the position of the current view's viewport in normalized screen coordinate space.

Host Syntax

```
EcRV first-corner
     second-corner
```

Setup Syntax

```
VIEWPORT first-corner
         second-corner
```

first-corner: xy-coordinate; specifies the location of one corner of the current view's viewport in normalized screen coordinates. Values for x must be in range 0 through 4095; y must be in range 0 through 3071.

Defaults:	Factory	= (0,0)
	Power-Up	= (0,0)
	Omitted	= (0,0)

second-corner: xy-coordinate; specifies location of the opposite corner of the viewport in normalized screen coordinates. Values for x must be in range 0 through 4095; y must be in range 0 through 3071.

Defaults:	Factory	= (4095,3071)
	Power-Up	= (4095,3071)
	Omitted	= (0,0)

The two xy-coordinates specify the positions of the lower-left and upper right corners of the viewport. (Actually, any two opposite corners will do; the terminal sorts the two x-coordinates and the two y-coordinates in the proper order.)

Even though the viewport exists in 640 by 480 raster memory space, the viewport corners are specified in 4096 by 3072 normalized screen coordinate space. The terminal internally converts these normalized screen coordinates to raster memory space coordinates.

NOTE

Changing the viewport changes the location on the screen where existing segments are displayed. However, segments that are visible when the viewport change occurs do not automatically move to their new screen locations. To make the terminal redraw segments at their new screen locations, you should issue a RENEW VIEW or PAGE command immediately after changing the viewport.

If you don't do this, and the fixup level is less than or equal to 4, then moving a segment may not cause that segment's old image to be properly removed from the screen. Multiple images of the segment will appear. The remedy is to issue RENEW VIEW or PAGE either immediately after the SET VIEWPORT command or immediately after moving a segment.

SET WINDOW

Sets the boundaries of the current view's 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 in terminal space for the current view. Values for x and y must be in the range 0 through 4095.

Defaults:	Factory	= (0,0)
	Power-Up	= Saved in memory
	Omitted	= (0,0)

second-corner: xy-coordinate; specifies the opposite corner of the window. Values for x and y must be in range 0 through 4095.

Defaults:	Factory	= (4095,3130)
	Power-Up	= Saved in Memory
	Omitted	= (0,0)

The window is the rectangular region in terminal space whose contents are displayed in a viewport on the screen.

The parameters 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.

Specifying a Window of Zero Width. If the rectangle specified has zero width (that is, if the the two x-coordinates are equal), then the window (in terminal space) will have the same aspect ratio (ratio of height to width) as the corresponding viewport (in raster memory space). In that window, x-min (the x-coordinate of the lower-left corner) is the x-coordinate specified in the command. The value for x-max (the x-coordinate of the upper-right corner) is automatically chosen so that the window is the same shape as the viewport.

Specifying a Window of Zero Height. Likewise, if the two corner parameters specify a rectangle of zero height (that is, if the two y-coordinates are equal), then the window (in terminal space) will have the same shape as the corresponding viewport (in raster memory space). In that window, y-min is the y-coordinate specified in the command's two parameters. The value for y-max is chosen so that the window has the same shape as the viewport.

Specifying the Default Window. If both width and height are zero (that is, if both lower-left and upper-right refer to the same point), then a window is selected which extends from x = 0 through x = 4095, and from y = 0 through y = 3136.

NOTE

Changing the window changes the location on the screen where existing segments are displayed. However, segments that are visible when the window change occurs do not automatically move to their new screen locations. To make the terminal redraw segments at their new screen locations, you should issue a RENEW VIEW or PAGE command immediately after changing the window.

If you don't do this, and the fixup level is greater than or equal to 4, then moving a segment may not cause that segment's old image to be properly removed from the screen. Multiple images of the segment will appear. The remedy is to issue RENEW VIEW or PAGE either immediately after the SET WINDOW command or immediately after moving a segment.

Effect On Other Views in the Same View Display Cluster. Views may be grouped into view display clusters. (See the SET VIEW DISPLAY CLUSTER command for details.) If the current view belongs to a view display cluster, and a SET WINDOW command is issued, then that SET WINDOW command sets the window not only for the current view, but also for all other views in that cluster.

Clipping. The terminal performs a window-viewport transform on each visible segment; that is, for each segment defined in terminal space coordinates, the terminal computes an image of that segment in 640 by 480 raster memory space. The window coordinates (in terminal space) and the viewport coordinates (in raster memory space) together define this window-viewport transform. Those parts of the image in raster memory space which fall outside the current viewport are clipped (not displayed).

SET 4014 ALPHATEXT SIZE

Selects between two alphatext character sizes.

Host Syntax

```
Ec size-code
```

size-code: specifies one of two sizes for alphatext. Must be one of four ASCII characters:

8 or 9	Fits at least 80 characters on one line
: or ;	Fits at least 128 characters on one line

This command allows the terminal to be compatible with earlier Tektronix terminals. It affects the terminal only when the dialog area is disabled.

When using the 128 character-per-line size the terminal displays characters only in the US ASCII font.

SET 4014 LINE STYLE

Specifies line styles compatible with Tektronix 4010 and 4110 Series terminals.

Host Syntax

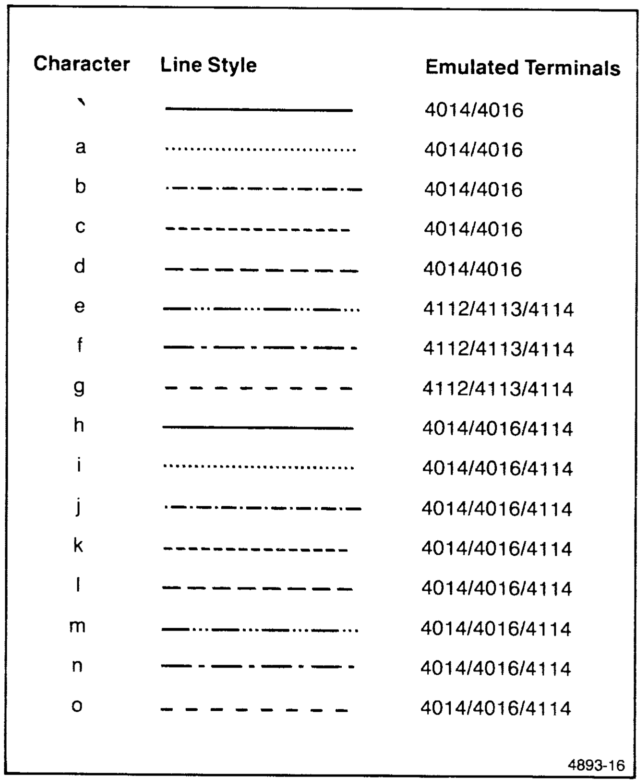
```
Ec line-style-code
```

line-style-code: single character; specifies one of the line styles shown in Figure 5-15.

This command does the same thing as the SET LINE STYLE command. The line style for lines, markers, and panel boundaries is set by the last 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 4107 and 4109 terminals does not defocus these lines.



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 5-15. Line Style Codes.

STATUS

Displays the current parameter values for a command or cluster of commands.**Setup Syntax**

```
STATUS name
```

name: string; the Setup 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** (from Setup mode).
Default: Omitted = All commands

4010 HARDCOPY

Generates a hardcopy of the entire screen.

Host Syntax

```
EC EB
```

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 use the dialog area while the copy is being made.

If you copy the dialog area, during the time the copy is being made, the dialog area will not show your dialog with the host.

Appendix A

CODE CHARTS AND KEYBOARD MACROS

Each code chart in this appendix contains one of the eight character sets available in the terminal. These character sets are defined in the ANSI X3.41 and ISO 2022 documents.

You can designate any two of the character sets as the G0 and G1 sets and easily switch between them. The G0 set is selected by sending a $^{S}{}_{I}$ (Shift In) character to the terminal, which causes the G0 character set to replace the current character set in the terminal. The G1 set is selected by sending a $^{S}{}_{O}$ (Shift Out) character to the terminal, which causes the G1 character set to replace the current G0 character set in the terminal. Refer to the description of the SCS (Select Character Set), $^{S}{}_{I}$ (Shift In), and $^{S}{}_{O}$ (Shift Out) commands in the *Screen Editing* section of this manual for more information.

Each optional keyboard has a character set that corresponds to its special characters. Plugging a keyboard into the terminal automatically selects this character set as both the G0 and G1 character sets. For example, the North American keyboard selects the ASCII character set, while the Option 4G German keyboard selects the German character set. However, this does not limit a keyboard to one character set, or limit a character set's use to one keyboard; all character sets are available for selection as the G0 or G1 set, regardless of which keyboard you attach to the terminal.

At the end of this appendix are the code charts for the Supplementary and Rulings character sets. These character sets are available to all keyboards, but none of the keyboards automatically selects either of them.

Each keyboard macro chart represents one of the six keyboards available with the terminal, and lists the macro numbers invoked by each of the keys and key combinations. Each keyboard macro chart accompanies the code chart of the keyboard's corresponding character set. More information about macros and how they are defined and used can be found in the section called *The Graphics Terminal*, and in the DEFINE MACRO command description.

ASCII CHARACTER SET

(Designated by Connecting the Standard North American Keyboard)

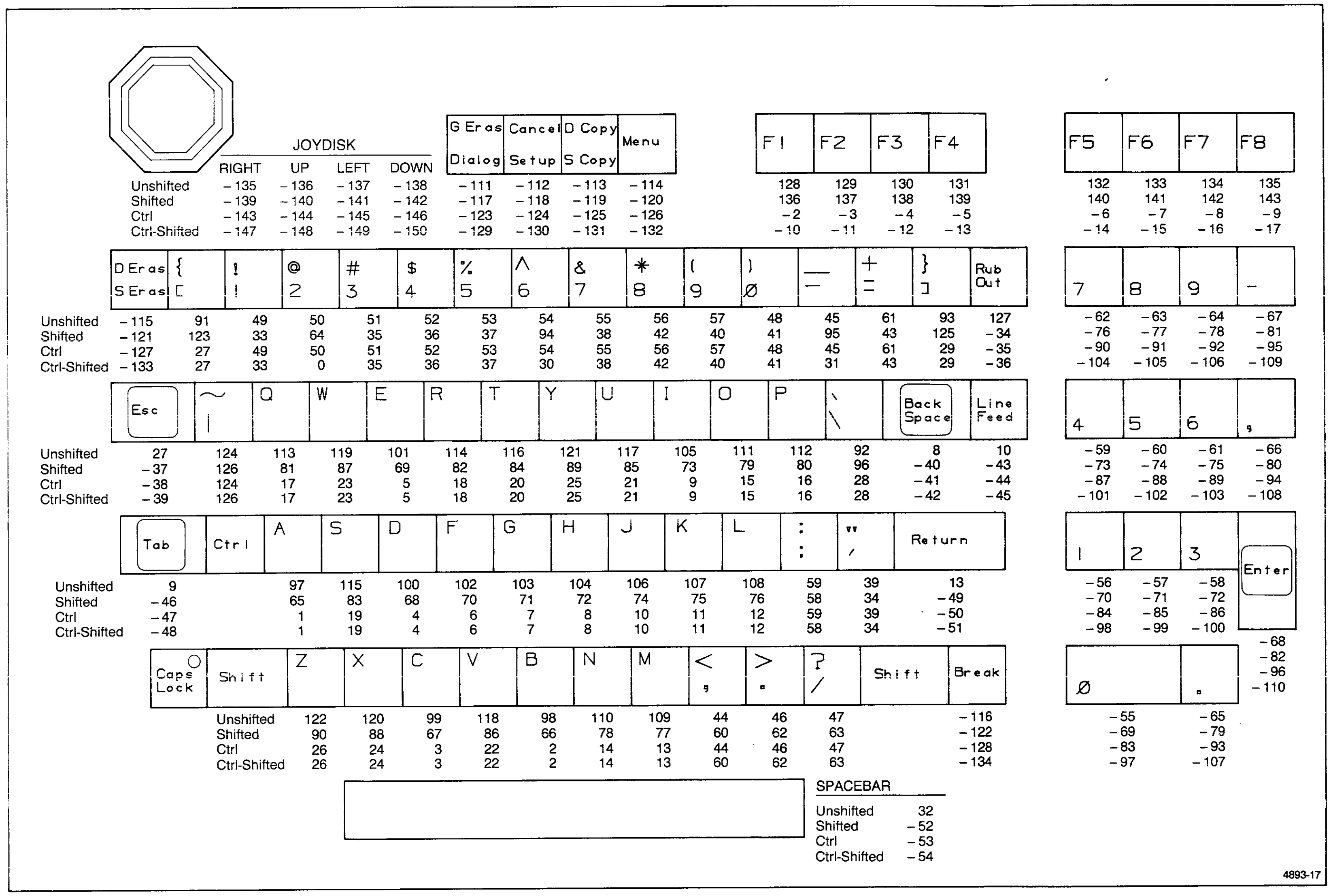
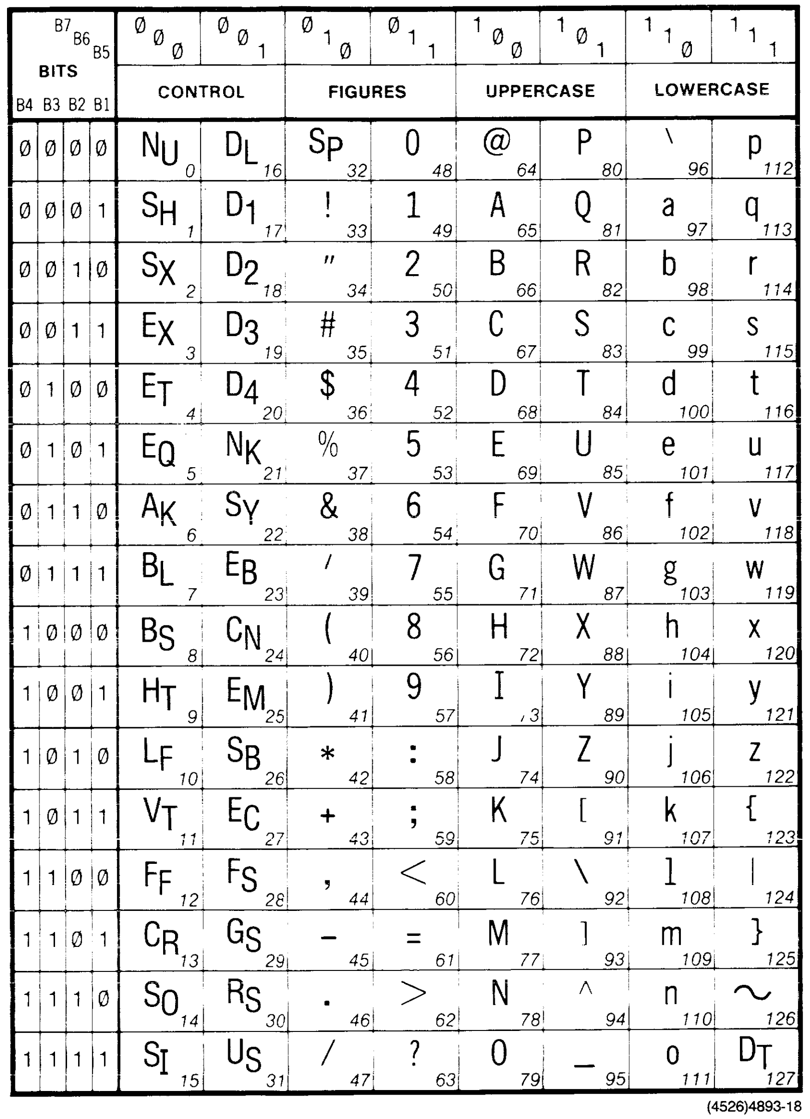


Figure A-1. North American Keyboard Macro Chart.



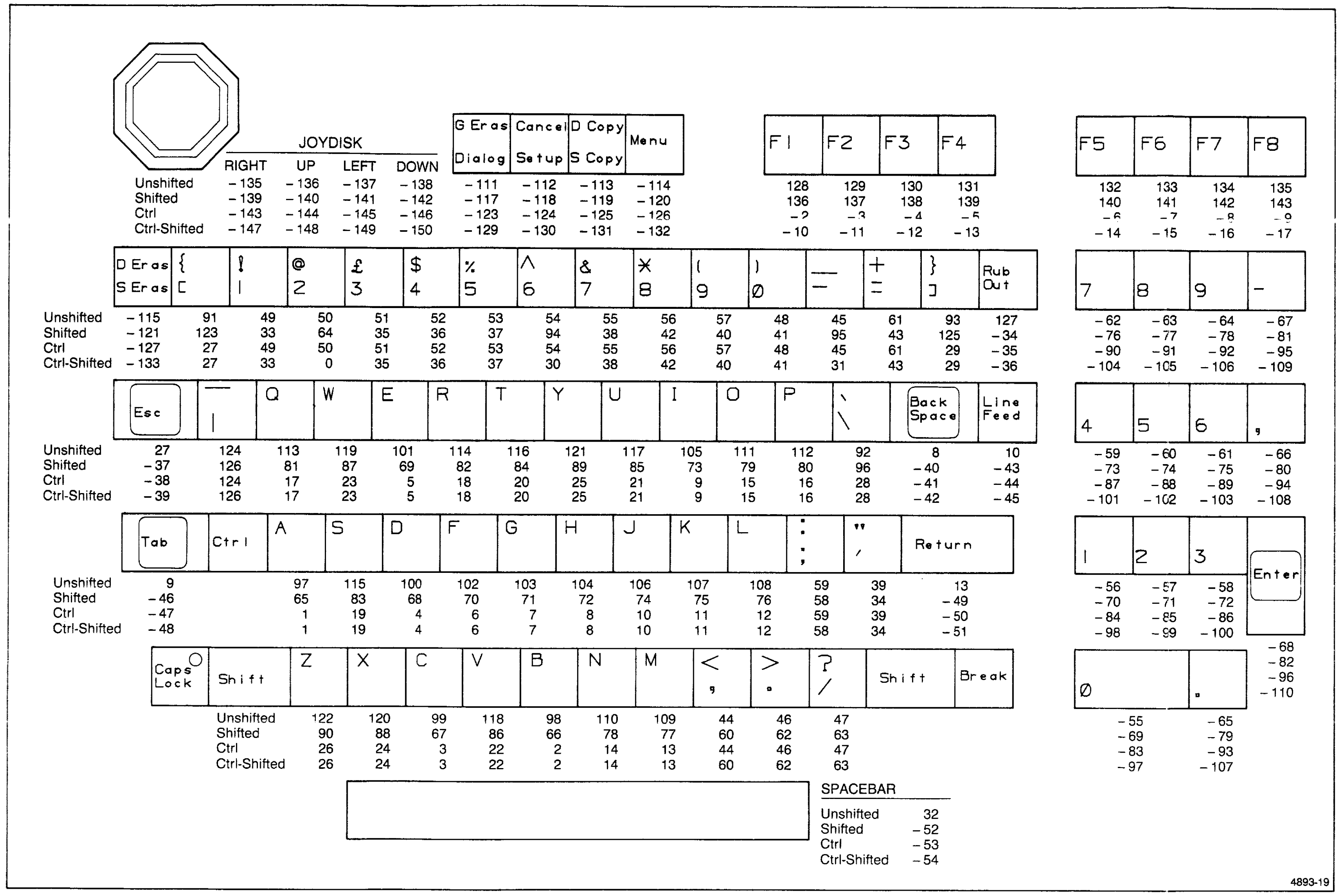
B7 B6 B5 BITS	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
B4 B3 B2 B1	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
Ø Ø Ø Ø	NU 0	DL 16	SP 32	0 48	@ 64	P 80	` 96	p 112
Ø Ø Ø 1	SH 1	D1 17	! 33	1 49	A 65	Q 81	a 97	q 113
Ø Ø 1 Ø	SX 2	D2 18	" 34	2 50	B 66	R 82	b 98	r 114
Ø Ø 1 1	EX 3	D3 19	# 35	3 51	C 67	S 83	c 99	s 115
Ø 1 Ø Ø	ET 4	D4 20	$ 36	4 52	D 68	T 84	d 100	t 116
Ø 1 Ø 1	EQ 5	NK 21	% 37	5 53	E 69	U 85	e 101	u 117
Ø 1 1 Ø	AK 6	SY 22	& 38	6 54	F 70	V 86	f 102	v 118
Ø 1 1 1	BL 7	EB 23	' 39	7 55	G 71	W 87	g 103	w 119
1 Ø Ø Ø	BS 8	CN 24	(40	8 56	H 72	X 88	h 104	x 120
1 Ø Ø 1	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1 Ø 1 Ø	LF 10	SB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1 Ø 1 1	VT 11	EC 27	+ 43	; 59	K 75	[91	k 107	{ 123
1 1 Ø Ø	FF 12	FS 28	, 44	< 60	L 76	\ 92	l 108	| 124
1 1 Ø 1	CR 13	GS 29	− 45	= 61	M 77	] 93	m 109	} 125
1 1 1 Ø	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-2. ASCII Character Set Code Chart

UNITED KINGDOM CHARACTER SET

(Designated by Connecting the Option 4A Keyboard)



	JOYDISK RIGHT	JOYDISK UP	JOYDISK LEFT	JOYDISK DOWN	G Eras / Dialog	Cancel / Setup	D Copy / S Copy	Menu	F1	F2	F3	F4	F5	F6	F7	F8
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 / S Eras	{ / [	! / 1	@ / 2	£ / 3	$ / 4	% / 5	^ / 6	& / 7	* / 8	(/ 9	) / Ø	— / –	+ / =	} /]	Rub Out	7	8	9	–
Unshifted	-115	91	49	50	51	52	53	54	55	56	57	48	45	61	93	127	-62	-63	-64	-67
Shifted	-121	123	33	64	35	36	37	94	38	42	40	41	95	43	125	-34	-76	-77	-78	-81
Ctrl	-127	27	49	50	51	52	53	54	55	56	57	48	45	61	29	-35	-90	-91	-92	-95
Ctrl-Shifted	-133	27	33	0	35	36	37	30	38	42	40	41	31	43	29	-36	-104	-105	-106	-109

	Esc	— / |	Q	W	E	R	T	Y	U	I	O	P	\` / \	Back Space	Line Feed	4	5	6	,
Unshifted	27	124	113	119	101	114	116	121	117	105	111	112	92	8	10	-59	-60	-61	-66
Shifted	-37	126	81	87	69	82	84	89	85	73	79	80	96	-40	-43	-73	-74	-75	-80
Ctrl	-38	124	17	23	5	18	20	25	21	9	15	16	28	-41	-44	-87	-88	-89	-94
Ctrl-Shifted	-39	126	17	23	5	18	20	25	21	9	15	16	28	-42	-45	-101	-102	-103	-108

	Tab	Ctrl	A	S	D	F	G	H	J	K	L	: / ;	" / '	Return	1	2	3	Enter
Unshifted	9		97	115	100	102	103	104	106	107	108	59	39	13	-56	-57	-58	-68
Shifted	-46		65	83	68	70	71	72	74	75	76	58	34	-49	-70	-71	-72	-82
Ctrl	-47		1	19	4	6	7	8	10	11	12	59	39	-50	-84	-85	-86	-96
Ctrl-Shifted	-48		1	19	4	6	7	8	10	11	12	58	34	-51	-98	-99	-100	-110

	Caps Lock	Shift	Z	X	C	V	B	N	M	< / ,	> / .	? / /	Shift	Break	Ø	.
Unshifted			122	120	99	118	98	110	109	44	46	47			-55	-65
Shifted			90	88	67	86	66	78	77	60	62	63			-69	-79
Ctrl			26	24	3	22	2	14	13	44	46	47			-83	-93
Ctrl-Shifted			26	24	3	22	2	14	13	60	62	63			-97	-107

SPACEBAR	
Unshifted	32
Shifted	-52
Ctrl	-53
Ctrl-Shifted	-54

4893-19

Figure A-3. United Kingdom Keyboard Macro Chart.

BITS B4 B3 B2 B1 \ B7 B6 B5	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
Ø Ø Ø Ø	NU 0	DL 16	SP 32	0 48	@ 64	P 80	\` 96	p 112
Ø Ø Ø 1	SH 1	D1 17	! 33	1 49	A 65	Q 81	a 97	q 113
Ø Ø 1 Ø	SX 2	D2 18	" 34	2 50	B 66	R 82	b 98	r 114
Ø Ø 1 1	EX 3	D3 19	# 35	3 51	C 67	S 83	c 99	s 115
Ø 1 Ø Ø	ET 4	D4 20	£ 36	4 52	D 68	T 84	d 100	t 116
Ø 1 Ø 1	EQ 5	NK 21	% 37	5 53	E 69	U 85	e 101	u 117
Ø 1 1 Ø	AK 6	SY 22	& 38	6 54	F 70	V 86	f 102	v 118
Ø 1 1 1	BL 7	EB 23	' 39	7 55	G 71	W 87	g 103	w 119
1 Ø Ø Ø	BS 8	CN 24	(40	8 56	H 72	X 88	h 104	x 120
1 Ø Ø 1	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1 Ø 1 Ø	LF 10	SB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1 Ø 1 1	VT 11	EC 27	+ 43	; 59	K 75	[91	k 107	{ 123
1 1 Ø Ø	FF 12	FS 28	, 44	< 60	L 76	\ 92	l 108	| 124
1 1 Ø 1	CR 13	GS 29	– 45	= 61	M 77	] 93	m 109	} 125
1 1 1 Ø	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)4893-20

Figure A-4. United Kingdom Character Set Code Chart

FRENCH CHARACTER SET

(Designated by Connecting the Option 4B Keyboard)

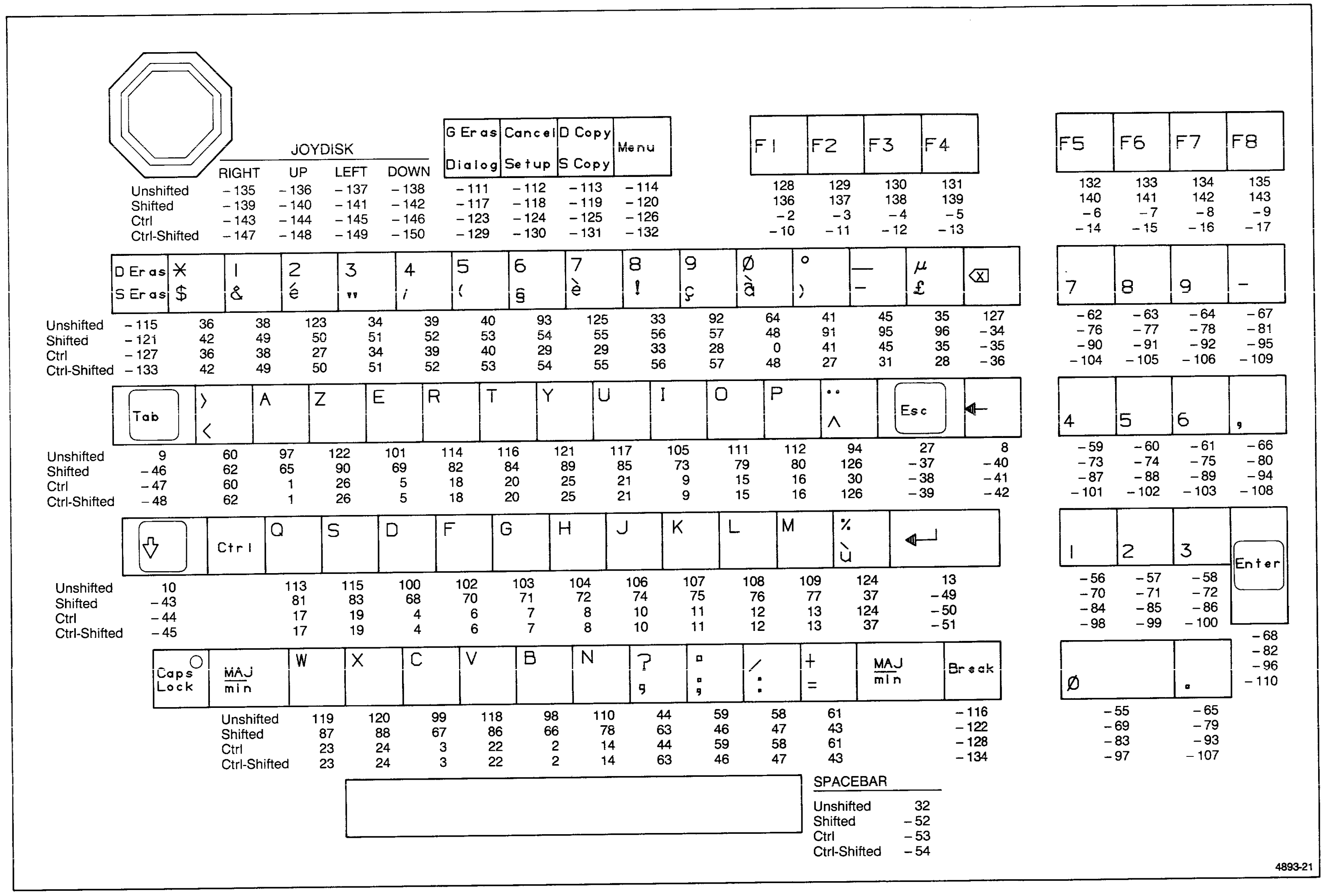

JOYDISK
RIGHT UP LEFT DOWN
Unshifted -135 -136 -137 -138
Shifted -139 -140 -141 -142
Ctrl -143 -144 -145 -146
Ctrl-Shifted -147 -148 -149 -150
G Eras / Dialog; Cancel / Setup; D Copy / S Copy; Menu
-111 -112 -113 -114
-117 -118 -119 -120
-123 -124 -125 -126
-129 -130 -131 -132
F1 F2 F3 F4
128 129 130 131
136 137 138 139
-2 -3 -4 -5
-10 -11 -12 -13
F5 F6 F7 F8
132 133 134 135
140 141 142 143
-6 -7 -8 -9
-14 -15 -16 -17
D Eras / S Eras; * $; 1 &; 2 é; 3 "; 4 '; 5 (; 6 §; 7 è; 8 !; 9 ç; Ø à; °); — -; µ £; ⌫
Unshifted -115 36 38 123 34 39 40 93 125 33 92 64 41 45 35 127
Shifted -121 42 49 50 51 52 53 54 55 56 57 48 91 95 96 -34
Ctrl -127 36 38 27 34 39 40 29 29 33 28 0 41 45 35 -35
Ctrl-Shifted -133 42 49 50 51 52 53 54 55 56 57 48 27 31 28 -36
7 8 9 -
-62 -63 -64 -67
-76 -77 -78 -81
-90 -91 -92 -95
-104 -105 -106 -109
Tab; > <; A; Z; E; R; T; Y; U; I; O; P; ¨ ^; Esc; ←
Unshifted 9 60 97 122 101 114 116 121 117 105 111 112 94 27 8
Shifted -46 62 65 90 69 82 84 89 85 73 79 80 126 -37 -40
Ctrl -47 60 1 26 5 18 20 25 21 9 15 16 30 -38 -41
Ctrl-Shifted -48 62 1 26 5 18 20 25 21 9 15 16 126 -39 -42
4 5 6 ,
-59 -60 -61 -66
-73 -74 -75 -80
-87 -88 -89 -94
-101 -102 -103 -108
⇩; Ctrl; Q; S; D; F; G; H; J; K; L; M; % ù; ↵
Unshifted 10 113 115 100 102 103 104 106 107 108 109 124 13
Shifted -43 81 83 68 70 71 72 74 75 76 77 37 -49
Ctrl -44 17 19 4 6 7 8 10 11 12 13 124 -50
Ctrl-Shifted -45 17 19 4 6 7 8 10 11 12 13 37 -51
1 2 3 Enter
-56 -57 -58 -68
-70 -71 -72 -82
-84 -85 -86 -96
-98 -99 -100 -110
Caps Lock; MAJ/min; W; X; C; V; B; N; ? ,; . ;; / :; + =; MAJ/min; Break
Unshifted 119 120 99 118 98 110 44 59 58 61 -116
Shifted 87 88 67 86 66 78 63 46 47 43 -122
Ctrl 23 24 3 22 2 14 44 59 58 61 -128
Ctrl-Shifted 23 24 3 22 2 14 63 46 47 43 -134
Ø .
-55 -65
-69 -79
-83 -93
-97 -107
SPACEBAR
Unshifted 32
Shifted -52
Ctrl -53
Ctrl-Shifted -54
4893-21

Figure A-5. French Keyboard Macro Chart.

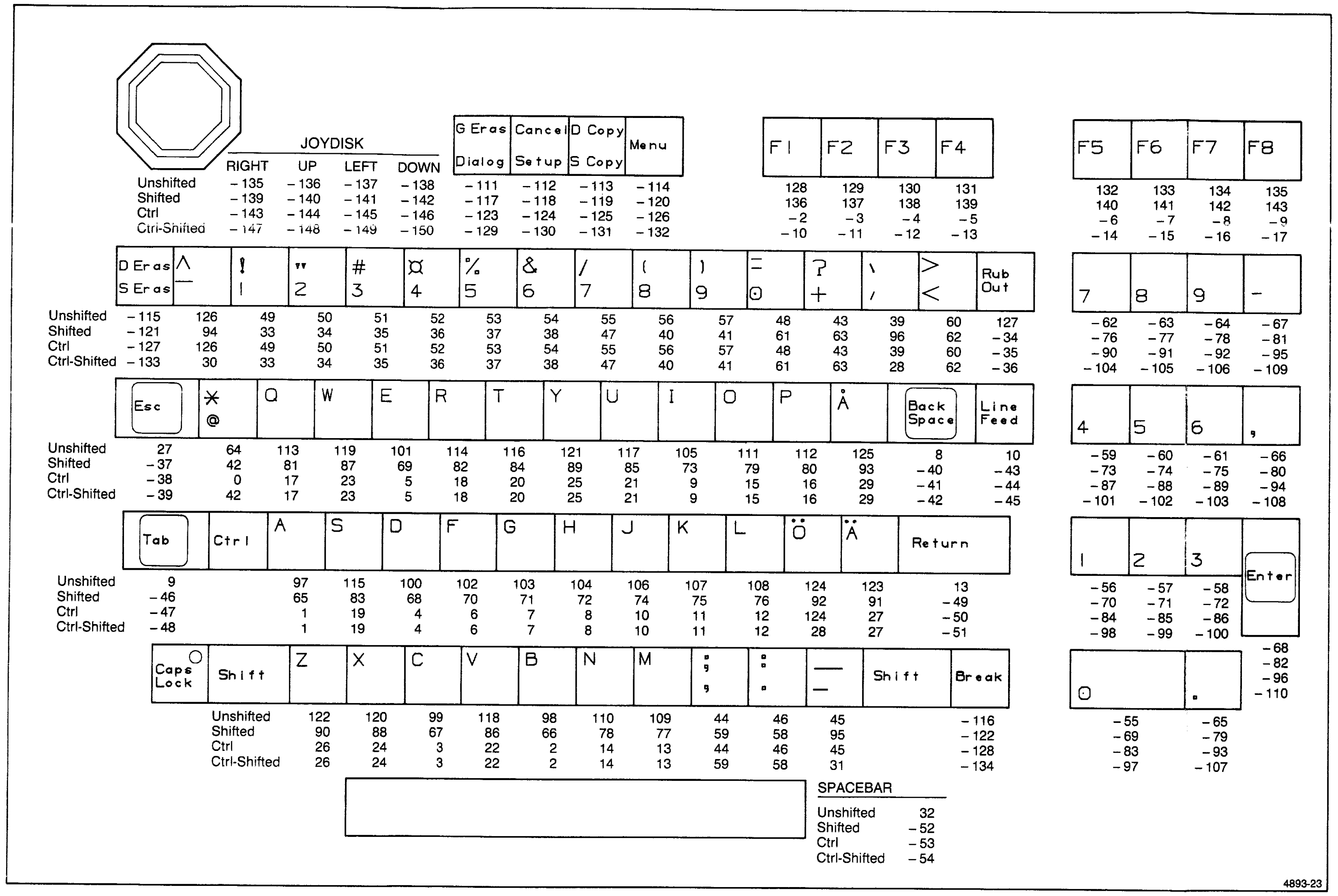
B7 B6 B5 → BITS				Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
B4	B3	B2	B1	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
Ø	Ø	Ø	Ø	NU 0	DL 16	SP 32	0 48	@ 64	P 80	\ 96	p 112
Ø	Ø	Ø	1	SH 1	D1 17	! 33	1 49	A 65	Q 81	a 97	q 113
Ø	Ø	1	Ø	SX 2	D2 18	" 34	2 50	B 66	R 82	b 98	r 114
Ø	Ø	1	1	EX 3	D3 19	£ 35	3 51	C 67	S 83	c 99	s 115
Ø	1	Ø	Ø	ET 4	D4 20	$ 36	4 52	D 68	T 84	d 100	t 116
Ø	1	Ø	1	EQ 5	NK 21	% 37	5 53	E 69	U 85	e 101	u 117
Ø	1	1	Ø	AK 6	SY 22	& 38	6 54	F 70	V 86	f 102	v 118
Ø	1	1	1	BL 7	EB 23	' 39	7 55	G 71	W 87	g 103	w 119
1	Ø	Ø	Ø	BS 8	CN 24	(40	8 56	H 72	X 88	h 104	x 120
1	Ø	Ø	1	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1	Ø	1	Ø	LF 10	SB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1	Ø	1	1	VT 11	EC 27	+ 43	; 59	K 75	° 91	k 107	é 123
1	1	Ø	Ø	FF 12	FS 28	, 44	< 60	L 76	ç 92	l 108	ù 124
1	1	Ø	1	CR 13	GS 29	- 45	= 61	M 77	§ 93	m 109	è 125
1	1	1	Ø	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)4893-22

Figure A-6. French Character Set Code Chart

SWEDISH CHARACTER SET

(Designated by Connecting the Option 4C Keyboard)



	RIGHT	UP	LEFT	DOWN	G Eras / Dialog	Cancel / Setup	D Copy / S Copy	Menu	F1	F2	F3	F4	F5	F6	F7	F8
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 / S Eras	^ / ‾	! / 1	" / 2	# / 3	¤ / 4	% / 5	& / 6	/ / 7	(/ 8	) / 9	= / 0	? / +	\ / ´	> / <	Rub Out	7	8	9	-
Unshifted	−115	126	49	50	51	52	53	54	55	56	57	48	43	39	60	127	−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
Ctrl	−127	126	49	50	51	52	53	54	55	56	57	48	43	39	60	−35	−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

	Esc	* / @	Q	W	E	R	T	Y	U	I	O	P	Å	Back Space	Line Feed	4	5	6	,
Unshifted	27	64	113	119	101	114	116	121	117	105	111	112	125	8	10	−59	−60	−61	−66
Shifted	−37	42	81	87	69	82	84	89	85	73	79	80	93	−40	−43	−73	−74	−75	−80
Ctrl	−38	0	17	23	5	18	20	25	21	9	15	16	29	−41	−44	−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

	Tab	Ctrl	A	S	D	F	G	H	J	K	L	Ö	Ä	Return	1	2	3	Enter
Unshifted	9		97	115	100	102	103	104	106	107	108	124	123	13	−56	−57	−58	−68
Shifted	−46		65	83	68	70	71	72	74	75	76	92	91	−49	−70	−71	−72	−82
Ctrl	−47		1	19	4	6	7	8	10	11	12	124	27	−50	−84	−85	−86	−96
Ctrl-Shifted	−48		1	19	4	6	7	8	10	11	12	28	27	−51	−98	−99	−100	−110

	Caps Lock	Shift	Z	X	C	V	B	N	M	; / ,	: / .	_ / -	Shift	Break	0	.
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

Figure A-7. Swedish Keyboard Macro Chart.

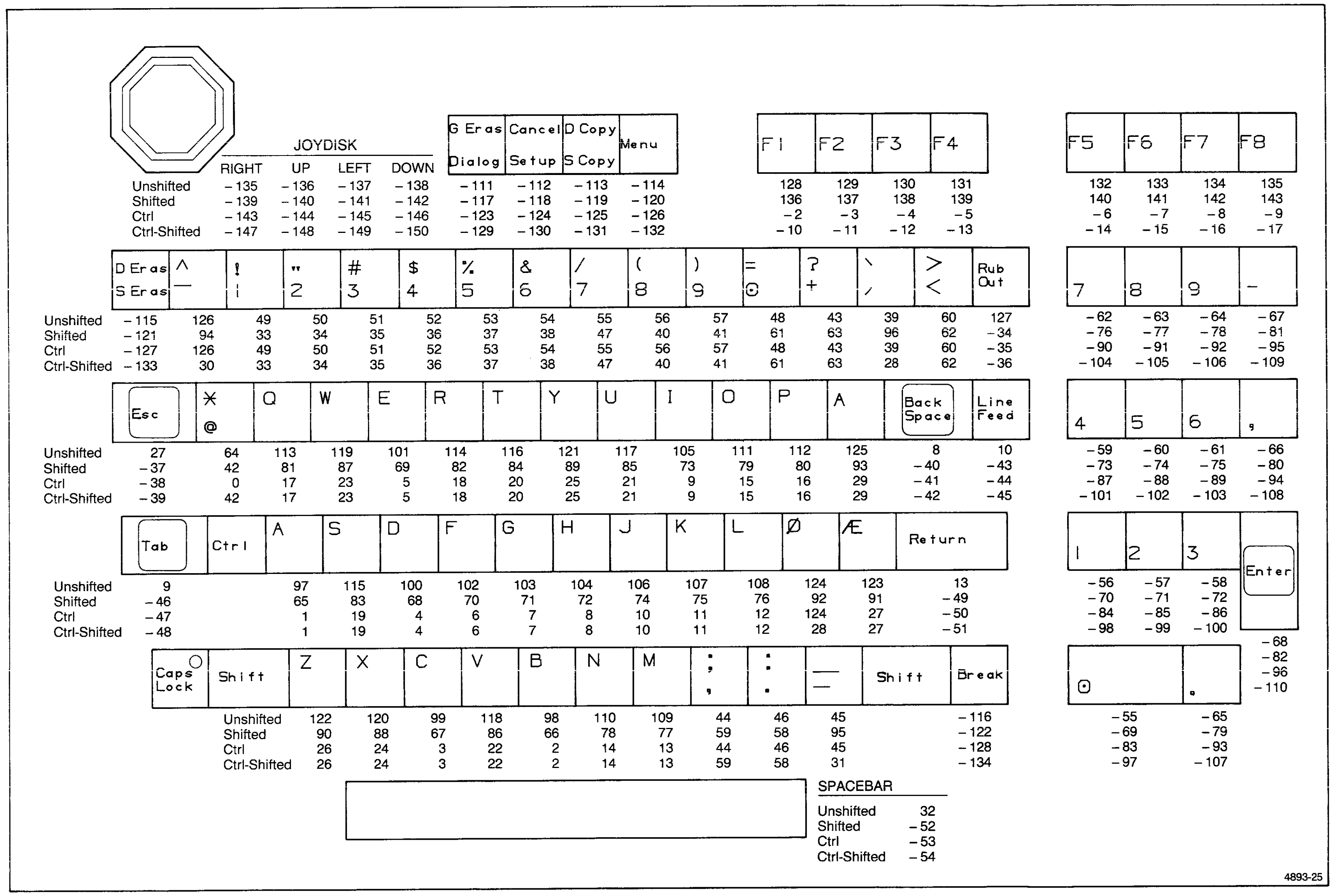
BITS B4 B3 B2 B1 \ B7 B6 B5	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
Ø Ø Ø Ø	NU 0	DL 16	SP 32	0 48	@ 64	P 80	\ 96	p 112
Ø Ø Ø 1	SH 1	D1 17	! 33	1 49	A 65	Q 81	a 97	q 113
Ø Ø 1 Ø	SX 2	D2 18	" 34	2 50	B 66	R 82	b 98	r 114
Ø Ø 1 1	EX 3	D3 19	# 35	3 51	C 67	S 83	c 99	s 115
Ø 1 Ø Ø	ET 4	D4 20	¤ 36	4 52	D 68	T 84	d 100	t 116
Ø 1 Ø 1	EQ 5	NK 21	% 37	5 53	E 69	U 85	e 101	u 117
Ø 1 1 Ø	AK 6	SY 22	& 38	6 54	F 70	V 86	f 102	v 118
Ø 1 1 1	BL 7	EB 23	' 39	7 55	G 71	W 87	g 103	w 119
1 Ø Ø Ø	BS 8	CN 24	(40	8 56	H 72	X 88	h 104	x 120
1 Ø Ø 1	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1 Ø 1 Ø	LF 10	SB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1 Ø 1 1	VT 11	EC 27	+ 43	; 59	K 75	Ä 91	k 107	ä 123
1 1 Ø Ø	FF 12	FS 28	, 44	< 60	L 76	Ö 92	l 108	ö 124
1 1 Ø 1	CR 13	GS 29	− 45	= 61	M 77	Å 93	m 109	å 125
1 1 1 Ø	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-8. Swedish Character Set Code Chart

DANISH/NORWEGIAN CHARACTER SET

(Designated by Connecting the Option 4F Keyboard)



	JOYDISK RIGHT	JOYDISK UP	JOYDISK LEFT	JOYDISK DOWN	G Eras / Dialog	Cancel / Setup	D Copy / S Copy	Menu
Unshifted	-135	-136	-137	-138	-111	-112	-113	-114
Shifted	-139	-140	-141	-142	-117	-118	-119	-120
Ctrl	-143	-144	-145	-146	-123	-124	-125	-126
Ctrl-Shifted	-147	-148	-149	-150	-129	-130	-131	-132

	F1	F2	F3	F4	F5	F6	F7	F8
Unshifted	128	129	130	131	132	133	134	135
Shifted	136	137	138	139	140	141	142	143
Ctrl	-2	-3	-4	-5	-6	-7	-8	-9
Ctrl-Shifted	-10	-11	-12	-13	-14	-15	-16	-17

	D Eras / S Eras	^ —	! 1	" 2	# 3	$ 4	% 5	& 6	/ 7	(8	) 9	= 0	? +	` ´	> <	Rub Out
Unshifted	-115	126	49	50	51	52	53	54	55	56	57	48	43	39	60	127
Shifted	-121	94	33	34	35	36	37	38	47	40	41	61	63	96	62	-34
Ctrl	-127	126	49	50	51	52	53	54	55	56	57	48	43	39	60	-35
Ctrl-Shifted	-133	30	33	34	35	36	37	38	47	40	41	61	63	28	62	-36

	7	8	9	-
Unshifted	-62	-63	-64	-67
Shifted	-76	-77	-78	-81
Ctrl	-90	-91	-92	-95
Ctrl-Shifted	-104	-105	-106	-109

	Esc	* @	Q	W	E	R	T	Y	U	I	O	P	Å	Back Space	Line Feed
Unshifted	27	64	113	119	101	114	116	121	117	105	111	112	125	8	10
Shifted	-37	42	81	87	69	82	84	89	85	73	79	80	93	-40	-43
Ctrl	-38	0	17	23	5	18	20	25	21	9	15	16	29	-41	-44
Ctrl-Shifted	-39	42	17	23	5	18	20	25	21	9	15	16	29	-42	-45

	4	5	6	,
Unshifted	-59	-60	-61	-66
Shifted	-73	-74	-75	-80
Ctrl	-87	-88	-89	-94
Ctrl-Shifted	-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
Shifted	-46		65	83	68	70	71	72	74	75	76	92	91	-49
Ctrl	-47		1	19	4	6	7	8	10	11	12	124	27	-50
Ctrl-Shifted	-48		1	19	4	6	7	8	10	11	12	28	27	-51

	1	2	3	Enter
Unshifted	-56	-57	-58	-68
Shifted	-70	-71	-72	-82
Ctrl	-84	-85	-86	-96
Ctrl-Shifted	-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
Shifted			90	88	67	86	66	78	77	59	58	95		-122
Ctrl			26	24	3	22	2	14	13	44	46	45		-128
Ctrl-Shifted			26	24	3	22	2	14	13	59	58	31		-134

	0	.
Unshifted	-55	-65
Shifted	-69	-79
Ctrl	-83	-93
Ctrl-Shifted	-97	-107

SPACEBAR	
Unshifted	32
Shifted	-52
Ctrl	-53
Ctrl-Shifted	-54

Figure A-9. Danish/Norwegian Keyboard Macro Chart.

BITS B7 B6 B5 / B4 B3 B2 B1	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
Ø Ø Ø Ø	NU 0	DL 16	SP 32	0 48	@ 64	P 80	\ 96	p 112
Ø Ø Ø 1	SH 1	D1 17	! 33	1 49	A 65	Q 81	a 97	q 113
Ø Ø 1 Ø	SX 2	D2 18	" 34	2 50	B 66	R 82	b 98	r 114
Ø Ø 1 1	EX 3	D3 19	# 35	3 51	C 67	S 83	c 99	s 115
Ø 1 Ø Ø	ET 4	D4 20	$ 36	4 52	D 68	T 84	d 100	t 116
Ø 1 Ø 1	EQ 5	NK 21	% 37	5 53	E 69	U 85	e 101	u 117
Ø 1 1 Ø	AK 6	SY 22	& 38	6 54	F 70	V 86	f 102	v 118
Ø 1 1 1	BL 7	EB 23	' 39	7 55	G 71	W 87	g 103	w 119
1 Ø Ø Ø	BS 8	CN 24	(40	8 56	H 72	X 88	h 104	x 120
1 Ø Ø 1	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1 Ø 1 Ø	LF 10	SB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1 Ø 1 1	VT 11	EC 27	+ 43	; 59	K 75	Æ 91	k 107	æ 123
1 1 Ø Ø	FF 12	FS 28	, 44	< 60	L 76	Ø 92	l 108	ø 124
1 1 Ø 1	CR 13	GS 29	- 45	= 61	M 77	Å 93	m 109	å 125
1 1 1 Ø	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-10. Danish/Norwegian Character Set Code Chart

GERMAN CHARACTER SET

(Designated by Connecting the Option 4G Keyboard)

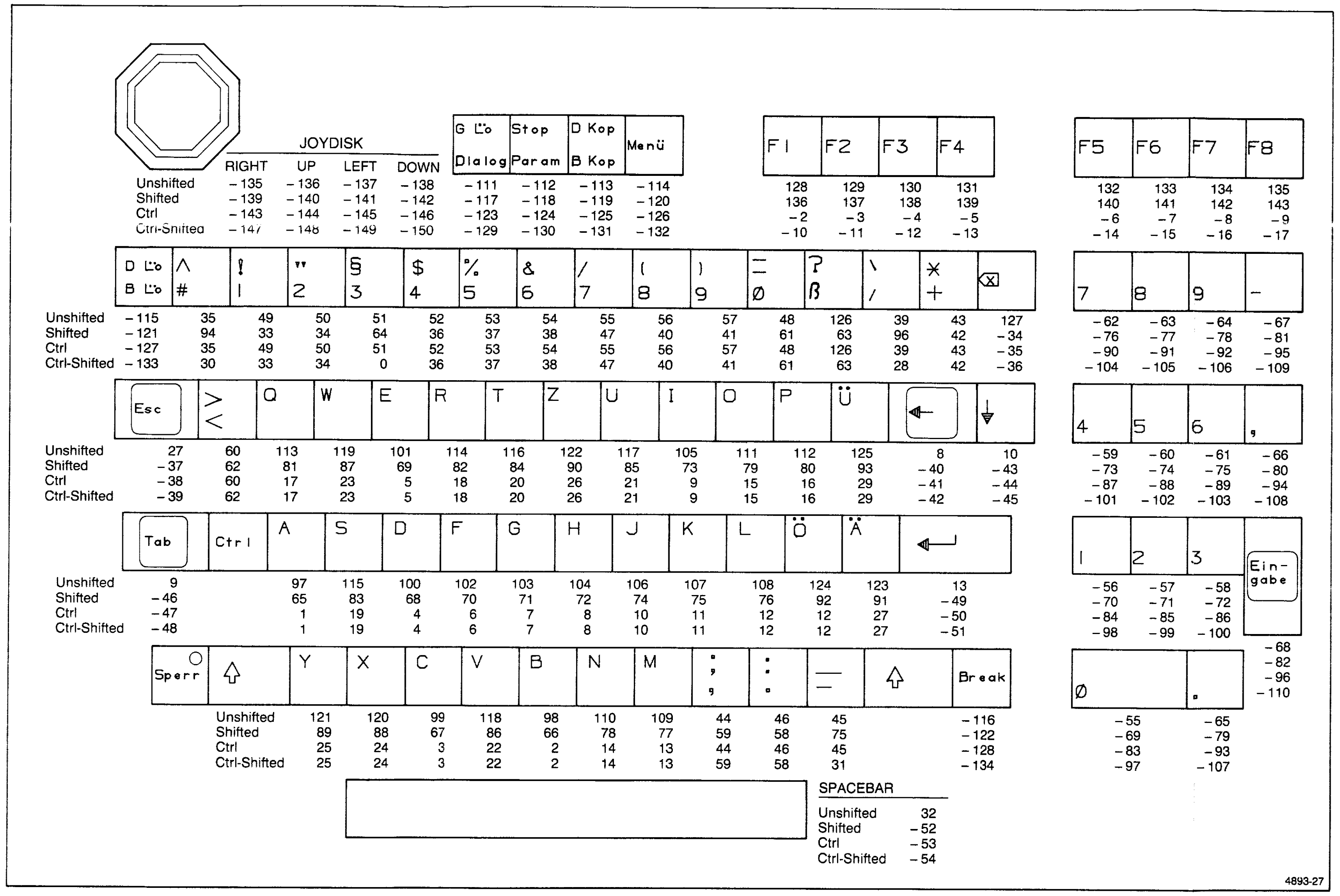


Figure A-11. German Keyboard Macro Chart.

B7 B6 B5 BITS / B4 B3 B2 B1	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
Ø Ø Ø Ø	NU 0	DL 16	SP 32	0 48	§ 64	P 80	\ 96	p 112
Ø Ø Ø 1	SH 1	D1 17	! 33	1 49	A 65	Q 81	a 97	q 113
Ø Ø 1 Ø	SX 2	D2 18	" 34	2 50	B 66	R 82	b 98	r 114
Ø Ø 1 1	EX 3	D3 19	# 35	3 51	C 67	S 83	c 99	s 115
Ø 1 Ø Ø	ET 4	D4 20	$ 36	4 52	D 68	T 84	d 100	t 116
Ø 1 Ø 1	EQ 5	NK 21	% 37	5 53	E 69	U 85	e 101	u 117
Ø 1 1 Ø	AK 6	SY 22	& 38	6 54	F 70	V 86	f 102	v 118
Ø 1 1 1	BL 7	EB 23	' 39	7 55	G 71	W 87	g 103	w 119
1 Ø Ø Ø	BS 8	CN 24	(40	8 56	H 72	X 88	h 104	x 120
1 Ø Ø 1	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1 Ø 1 Ø	LF 10	SB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1 Ø 1 1	VT 11	EC 27	+ 43	; 59	K 75	Ä 91	k 107	ä 123
1 1 Ø Ø	FF 12	FS 28	, 44	< 60	L 76	Ö 92	l 108	ö 124
1 1 Ø 1	CR 13	GS 29	- 45	= 61	M 77	Ü 93	m 109	ü 125
1 1 1 Ø	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)4893-28

Figure A-12. German Character Set Code Chart

SUPPLEMENTARY CHARACTER SET

B7 B6 B5 BITS				Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
B4	B3	B2	B1	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
Ø	Ø	Ø	Ø	NU 0	DL 16	SP 32	0 48	‾ 64	Ñ 80	◆ 96	▔ 112
Ø	Ø	Ø	1	SH 1	D1 17	Ä 33	1 49	¢ 65	ñ 81	█ 97	─ 113
Ø	Ø	1	Ø	SX 2	D2 18	ä 34	2 50	¦ 66	¿ 82	HT 98	─ 114
Ø	Ø	1	1	EX 3	D3 19	Å 35	3 51	† 67	¡ 83	FF 99	▁ 115
Ø	1	Ø	Ø	ET 4	D4 20	å 36	4 52	□ 68	α 84	CR 100	├ 116
Ø	1	Ø	1	EQ 5	NK 21	Æ 37	5 53	■ 69	σ 85	LF 101	┤ 117
Ø	1	1	Ø	AK 6	SY 22	æ 38	6 54	● 70	τ 86	° 102	┴ 118
Ø	1	1	1	BL 7	EB 23	à 39	7 55	Δ 71	ρ 87	± 103	┬ 119
1	Ø	Ø	Ø	BS 8	CN 24	ç 40	8 56	δ 72	μ 88	NL 104	│ 120
1	Ø	Ø	1	HT 9	EM 25	é 41	9 57	λ 73	Σ 89	VT 105	⩽ 121
1	Ø	1	Ø	LF 10	SB 26	è 42	ù 58	┐ 74	Ω 90	┘ 106	⩾ 122
1	Ø	1	1	VT 11	EC 27	Ö 43	β 59	└ 75	⌠ 91	┐ 107	π 123
1	1	Ø	Ø	FF 12	FS 28	ö 44	<θ 60	┌ 76	⌡ 92	┌ 108	≠ 124
1	1	Ø	1	CR 13	GS 29	φ 45	¤ 61	┘ 77	÷ 93	└ 109	£ 125
1	1	1	Ø	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)4893-29

Figure A-13. Supplementary Character Set Code Chart.

RULINGS CHARACTER SET

B7 B6 B5 BITS	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
B4 B3 B2 B1	CONTROL		FIGURES		UPPERCASE		LOWERCASE	
Ø Ø Ø Ø	NU 0	DL 16	SP 32	0 48	@ 64	P 80	◆ 96	⎺ 112
Ø Ø Ø 1	SH 1	D1 17	! 33	1 49	A 65	Q 81	█ 97	⎻ 113
Ø Ø 1 Ø	SX 2	D2 18	" 34	2 50	B 66	R 82	HT 98	⎼ 114
Ø Ø 1 1	EX 3	D3 19	# 35	3 51	C 67	S 83	FF 99	⎽ 115
Ø 1 Ø Ø	ET 4	D4 20	$ 36	4 52	D 68	T 84	CR 100	├ 116
Ø 1 Ø 1	EQ 5	NK 21	% 37	5 53	E 69	U 85	LF 101	┤ 117
Ø 1 1 Ø	AK 6	SY 22	& 38	6 54	F 70	V 86	° 102	┴ 118
Ø 1 1 1	BL 7	EB 23	' 39	7 55	G 71	W 87	± 103	┬ 119
1 Ø Ø Ø	BS 8	CN 24	(40	8 56	H 72	X 88	NL 104	│ 120
1 Ø Ø 1	HT 9	EM 25	) 41	9 57	I 73	Y 89	VT 105	⩽ 121
1 Ø 1 Ø	LF 10	SB 26	* 42	: 58	J 74	Z 90	┘ 106	⩾ 122
1 Ø 1 1	VT 11	EC 27	+ 43	; 59	K 75	[91	┐ 107	π 123
1 1 Ø Ø	FF 12	FS 28	, 44	< 60	L 76	\ 92	┌ 108	≠ 124
1 1 Ø 1	CR 13	GS 29	− 45	= 61	M 77	] 93	└ 109	£ 125
1 1 1 Ø	SO 14	RS 30	. 46	> 62	N 78	^ 94	┼ 110	· 126
1 1 1 1	SI 15	US 31	/ 47	? 63	O 79	95	▔ 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 for the operator. 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 (ERROR LEVEL 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*, an abbreviated code that 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:

$^{E}{}_{C}$**MP** fill-pattern-number

The "1" 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.** Errors of severity Level 0 are minor errors. The corresponding message begins with, "Terminal issues message . . . ."
- **Level 1.** Level 1 errors are warnings. The corresponding message begins with, "Terminal issues warning . . . ." 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 . . . ." 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 . . . ." For instance, there may be insufficient memory to hold all the information being entered.

ERROR CODES

The rest of this appendix lists each error code alphabetically with its severity level and an explanation of the cause.

I011 (Level 2). Invalid *device-function* code. (See the description of the ENABLE GIN command for a table of *device-function* codes.)

IA11 (Level 2). Invalid *aperture-width* (must range from 0 to 4095).

IC13 (Level 2). Graphic input has already been enabled for the specified *device-function-code.*

IC20 (Level 2). Segment does not exist, or is currently being defined.

IC21 (Level 2). Invalid *segment-number* (must range from 0 to 32767).

IE00 (Level 2). The cursor segment for the specified *device-function-code* does not exist. (It has been deleted after the SET GIN CURSOR command which assigned it to that *device-function-code.*)

IE03 (Level 2). Command is invalid at this time. (The segment being used as the cursor for the specified *device-function-code* is a segment which is currently being defined.)

IE10 (Level 2). The specified GIN device is not installed in the terminal.

IE13 (Level 2). The specified GIN device is already enabled.

IE21 (Level 2). Invalid *number-of-GIN-events.* (Must range from 0 to 65535.)

IF10 (Level 2). Stroke filtering is not valid for the specified *device-function-code.* (Only allowed for stroke function.)

IF21 (Level 2). Invalid *distance-filter* (range is 0 to 32767).

IF31 (Level 2). Invalid *time-filter* (range is 0 to 32767).

IG10 (Level 2). Gridding does not apply to the specified *device-function-code.* (Gridding is not allowed for the stroke function.)

IG21 (Level 2). Invalid *x-grid-spacing.* Must be in range 0 to 4095.

IG31 (Level 2). Invalid *y-grid-spacing.* Must be in range 0 through 4095.

II10 (Level 2). Inking does not apply to the specified *device-function-code.* (Inking is not allowed for the pick function.)

II21 (Level 2). Invalid inking mode (must be 0, 1 or 2).

IL11 (Level 2). Invalid *max-line-length.* (Must range from 0 to 65535.)

IM11 (Level 2). Invalid *EOM-frequency* setting (must be 0 or 1).

IP10 (Level 2). The specified GIN device is not installed in the terminal.

IP13 (Level 2). The *device-function-code* names a device which has already been enabled for a different graphic input function.

IR10 (Level 2). Rubberbanding does not apply to the specified *device-function-code.* (Rubberbanding is only allowed for the locator function. It is not allowed for the pick and stroke functions.)

IR21 (Level 2). Invalid rubberbanding mode (must be 0, 1, or 2).

IS21 (Level 2). Invalid *sig-char.* (Must range from 0 to 127).

IS31 (Level 2). Invalid *term-sig-char.* (Must range from 0 to 127.)

IV21 (Level 2). Invalid *window-specifier* (range is -1 to 64).

IV31 (Level 2). Invalid *first-corner* (X or Y out of range 0 to 4095).

IV41 (Level 2). Invalid *second-corner* (X or Y out of range 0 to 4095, or area is of zero width or height).

IW11 (Level 2). Invalid *lower-left-corner* (X or Y out of range 0 through 4095).

IW21 (Level 2). Invalid ~ upper-right corner (X or Y out of range 0 through 4095).

IX10 (Level 2). Invalid operation for define-function (see ENABLE GIN command).

IX21 (Level 2). Invalid *start-point* (x and y must be in the range 0 through 4095.

JC11 (Level 2). Invalid source device specifier (must be "HO:").

JC12 (Level 3). Out of memory while parsing the first parameter.

JC21 (Level 2). Invalid separator string (must be a ^{N}U character or the string "TO").

JC22 (Level 3). Out of memory while parsing the second parameter.

JC31 (Level 2). Invalid destination device specifier. Must be the string "HC:", "P0:", or "P1:"; or port is busy.

JC32 (Level 3). Out of memory while parsing the third parameter.

JC39 (Level 2). Hardcopy device not ready. Check the hard-copy unit.

KA03 (Level 1). Context problem, cannot disable dialog area when in ANSI mode.

KA11 (Level 2). Parameter out of range (must be 0 or 1).

KB12 (Level 2). Out of memory while parsing the array.

KD11 (Level 2). Invalid macro number (must be in range –150 to 32767).

KD21 (Level 2). Invalid character code in the ASCII-decimal-equivalents 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.

KE11 (Level 2). Invalid echo mode (must be 0 or 1).

KF11 (Level 2). Invalid LFCR mode (must be 0 or 1).

KH11 (Level 2). Invalid hardcopy code (must be 0, 1, or 2).

KI11 (Level 2). Invalid ignore-deletes mode (must be 0 or 1).

KL11 (Level 2). Invalid keyboard-lock mode (must be 0 or 1).

KO11 (Level 2). Invalid macro number.

KO21 (Level 2). Invalid character-code in the ADE parameter (character codes must be in teh range 0 through 127; the array count must be in the range 0 through 65535).

KO22 (Level 3). Insufficient memory to define macro.

KR11 (Level 2). Invalid "CR-implies-LF" mode (must be 0 or 1).

KS11 (Level 2). Invalid parameter (must be 0 or 1)

KT11 (Level 2). Invalid error threshold (must be in range from 0 to 4).

KU02 (Level 2). Nonvolatile memory hardware error.

KW11 (Level 2). Invalid mode (must be 0 or 1).

KX01 (Level 2). The maximum nesting depth for the EXPAND MACRO command has been exceeded. The nesting depth should not exceed five. Greater nesting depths may result in KX01 errors.

KX02 (Level 3). Out of memory while performing EXPAND MACRO command.

KX11 (Level 2). Invalid macro identifier (must be in the range from –150 to 32767).

KY11 (Level 2). Invalid key-execute-delimiter code (must be in range 0 to 127).

KZ11 (Level 2). Invalid *character-delete* character (must range from 0 to 127).

KZ21 (Level 2). Invalid *line-delete* character (must range from 0 to 127).

KZ31 (Level 2). Invalid *take-literally* character (must range from 0 to 127).

LB03 (Level 0). Dialog parameters modified.

LB11 (Level 2). Invalid number-of-lines parameter (must be in the range from 2 to 49).

LE02 (Level 3). Out of memory while performing END PANEL command.

LE03 (Level 1). No panel is currently being defined.

LI11 (Level 2). Invalid character index (must be in the range from 0 to 65535).

LI21 (Level 2). Invalid character background index (must be in the range from 0 to 65535).

LI31 (Level 2). Invalid dialog area wipe index (must be in the range from 0 to 65535).

LK02 (Level 3). Out of memory while performing INCLUDE COPY OF SEGMENT.

LK10 (Level 2). Segment does not exist.

LK11 (Level 2). Invalid *segment-number* (must be -3, -1, or in the range 1 through 32767).

LK13 (Level 2). The segment specified is currently being defined.

LL11 (Level 2). Invalid parameter (must be in the range of 2 through 30.

LM11 (Level 2). Invalid writing mode (must be 0 or 1).

LP03 (Level 2). Alphatext is not allowed within a PANEL DEFINITION. When the terminal is in alpha mode, be sure the dialog area is enabled; otherwise, the terminal attempts to read alphatext as xy parameters.

LP21 (Level 2). Invalid "draw boundary" mode (must be 0 or 1).

LT03 (Level 2). Command is invalid at this time. Graphtext is not allowed within a PANEL DEFINITION. When this error is detected, the panel being defined is closed, as if an END PANEL command were received.

LT11 (Level 2). Invalid graphtext string. Invalid array count (must be in range from 0 to 32767), or invalid character in the array (must be in the range from SP to (~) — decimal equivalents from 32 to 126).

LT12 (Level 3). Parameter 1 memory error (out of memory while parsing the string parameter).

LV11 (Level 2). Invalid dialog area visibility mode (must be 0 or 1).

MA11 (Level 2). Invalid *slant-angle* (must range from -32767.0 to +32767.0).

MB11 (Level 2). Invalid *text-background-index* (must range from -2 to 32767).

MB21 (Level 2). Invalid *dash-gap-index* (must be in range from -2 to 32767).

MC11 (Level 2). Invalid value in parameter 1 (must be in range 1 through 4095).

MC21 (Level 2). Invalid value in parameter 2 (must be in range 1 through 4095).

MC31 (Level 2). Invalid value in parameter 3 (must be in range 1 through 4095).

MF10 (Level 2). Font is not defined.

MF11 (Level 2). Invalid *font-number* (must range from 0 to 32767).

MG11 (Level 2). Invalid parameter (must be 0 or 1).

MI03 (Level 2). Command is invalid at this time (no segment is currently being defined).

MI11 (Level 2). Invalid *pick-number* (must range from 0 to 32767).

ML11 (Level 2). Invalid line index (must be in the range from 0 to 32767).

MM11 (level 2). Invalid marker type (must be in the range from 0 to 10)

MN11 (Level 2). Invalid rotation angle (must be either 0, 90, 180, or 270).

MP10 (Level 2). Specified fill pattern does not exist.

MP11 (Level 2). Invalid fill pattern number (must be in the range from -32768 to 16 or 50 to 174).

MQ11 (Level 2). Invalid *precision* mode (must be 1 or 2).

MR11 (Level 2). Invalid rotation angle (must be in the range −32767.0 to 32767.0).

MT11 (Level 2). Invalid text index (must be in the range from 0 to 65535).

MV11 (Level 2). Invalid line style (must be in the range from 0 to 7).

NB11 (Level 2). Invalid number of stop bits (must be 1 or 2).

NC11 (Level 2). Invalid value in parameter 1. Can be any ASCII character except the ^{N}U character.

NC21 (Level 2). Invalid value in parameter 2. Can be any ASCII character except the ^{N}U character.

ND11 (Level 2). Invalid transmit delay time (must be in the range from 0 to 65535 milliseconds).

NE11 (Level 2). Invalid EOF-string (must contain from 0 to 10 characters, with each character represented by an integer in the range from 0 to 127).

NE12 (Level 3). Parameter 1 memory error (out of memory while parsing the end-of-file-string parameter).

NF11 (Level 2). Invalid flagging mode (must be in the range from 0 to 4).

NK11 (Level 2). Invalid parameter (must be in the range from 0 to 65535).

NL11 (Level 2). Invalid transmit rate limit (must be in the range from 75 to 19200).

NM11 (Level 2). Invalid prompt mode parameter (must be 0, 1,or 2).

NP11 (Level 2). Invalid parity code (must be in the range from 0 to 4).

NQ02 (Level 3). Out of memory while performing SET QUEUE SIZE command.

NQ11 (Level 2). Invalid queue size (must be in the range from 1 to 65535).

NR11 (Level 2). Invalid transmit (terminal-to-host) data rate (must be either 1, 75, 110, 134, 150, 300, 600, 1200, 2400, 4800, 9600, or 19200).

NR21 (Level 2). Invalid receive (host-to-terminal) data rate (must be either 0, 1, 75, 110, 134, 150, 300, 600, 1200, 2400, 4800, 9600, or 19200).

NS11 (Level 2). Invalid prompt-string parameter (must be an array of up to 10 ASCII characters).

NS12 (Level 3). Parameter 1 memory error (out of memory while parsing the prompt-string parameter).

NT11 (Level 2). Invalid array count in EOL-string. The array must hold from 0 to 2 int parameters. Each int in the array must be in the range from 0 to 127.

NT12 (Level 3). Parameter 1 memory error (out of memory while parsing the EOL-string parameter).

NU11 (Level 2). Invalid numeric equivalent of bypass-cancel character (must be in the range from 0 to 255).

PA00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

PA11 (Level 2). Invalid *port-identifier* (must be **P0:** or **P1:**).

PA12 (Level 3). Out of memory while parsing the *port-identifier* string.

PA21 (Level 2). Invalid *protocol-identifier* (must be **PPORT**, **4662**, **4662/MP**, **4663**, or **4681**).

PA22 (Level 3). Out of memory while parsing *protocol-identifier* string.

PB00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

PB11 (Level 2). Invalid *port-name* (must be **P0:** or **P1:**).

PB12 (Level 3). Out of memory while parsing the *port-name* string.

PB21 (Level 2). Invalid *number-of-stop-bits* (must be 1 or 2).

PB31 (Level 2). Invalid *number-of-data-bits* (must be 5, 6, 7, or 8).

PE00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

PE11 (Level 2). Invalid *port-name* (must be **P0:** or **P1:**).

PE12 (Level 3). Out of memory while parsing the *port-name* string.

PE21 (Level 2). Invalid *EOF-string* (the integer array must have from 0 to 10 elements, and each integer in the array must be in the range from 0 to 127.)

PE22 (Level 3). Out of memory while parsing the *EOF-string* integer array.

PF11 (Level 2). Invalid *port-name* (must be **P0:** or **P1:**).

PF12 (Level 3). Out of memory while parsing the *port-name* string.

PF21 (Level 2). Invalid *flagging-mode* (must be 0, 1 or 2).

PF31 (Level 2). Invalid **GO** character (must be in range 0 to 127).

PF41 (Level 2). Invalid **STOP** character (must be in range 0 to 127; if non-zero, must be different from the **GO** character).

PI00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

PI11 (Level 2). Invalid *port-name* (must be **P0:** or **P1:**).

PI12 (Level 2). Out of memory while parsing the *port-name* string.

PI21 (Level 2). Invalid *index* (must be in the range −1 through 255).

PI31 (Level 2). Invalid *pen-number* (must be in the range 0 through 255).

PL00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

PL11 (Level 2). Invalid *separator*. (Must be the empty string or **TO**.)

PL12 (Level 3). Out of memory while parsing the *separator-string* .

PL21 (Level 2). Invalid *output-specifier* (must be **P0:** or **P1:**), or port is busy.

PL22 (Level 3). Out of memory while parsing the *output-specifier.*

PL23 (Level 2). Parameter 2 context error (invalid destination device, or device is busy.)

PP11 (Level 2). Invalid *port-name* (must be **P0:** or **P1:**), or port is busy.

PP12 (Level 3). Out of memory while parsing the *port-name* string.

PP21 (Level 2). Invalid *parity-mode* (must be in range 0 to 4).

PQ00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

PQ11 (Level 2). Invalid *port-name* (must be **P0:** or **P1:**).

PQ12 (Level 3). Out of memory while parsing the *port-name* string.

PR00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

PR11 (Level 2). Invalid *port-name* (must be **P0:** or **P1:**).

PR12 (Level 3). Out of memory while parsing the *port-name* string.

PR21 (Level 2). Invalid *baud-rate* (must be 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 4800, or 9600).

QA11 (Level 2). Invalid copy size (must be 0 or 1).

QD11 (Level 2). Invalid copier value.

QL11 (Level 2). Invalid page number.

QL21 (Level 2). Invalid page origin.

QL31 (Level 2). Invalid *^{F}F-interpretation* value.

RA10 (Level 2). Surface does not exist (has not been defined with SET SURFACE DEFINITIONS command).

RA11 (Level 2). Invalid surface number (must be in the range −1 through 4).

RA21 (Level 2). Invalid background index (must be in range 0 through 65535).

RA31 (Level 2). Invalid value (must be in range 0 through 65535).

RC00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

RC11 (Level 2). Invalid *view-number* (must be in the range −1 through 64).

RD00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

RD10 (Level 2). Occupied undefined surface. (This command would have resulted in a dialog area viewport, pixel viewport, or numbered graphic viewport residing on an undefined surface.)

RD11 (Level 2). Invalid *surface-defs* array (there must be one to four integers in each array and each integer must be in the range 0 through 4).

RD12 (Level 3). Out of memory while parsing the *surface-defs* array.

RE00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

RE11 (Level 2). Invalid *border-visibility-mode* parameter (must be 0, 1, or 2).

RF00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

RF11 (Level 2). Invalid *fixup-level* (must be in the range 0 through 32767).

RI00 (Level 0): Unrecognized command (terminal is not a 4107 or 4109).

RI10 (Level 2). A surface in *surface-numbers-and-visibilities* does not exist (has not been defined with a SET SURFACE DEFINITIONS command).

RI11 (Level 2). Invalid *surface-numbers-and-visibilities* array (each surface number must be in the range 1 through 4; each visibility code must be 0, 1, or 2).

RI12 (Level 3). Out of memory while parsing the *surface-numbers-and-visibilities* array.

RJ11 (Level 2). Invalid *locking-mode* (must be 0 or 1).

RK10 (Level 1). The designated view does not exist (has not been defined with a SELECT VIEW command).

RK11 (Level 2). Invalid *view-number* (must be in the range −1 through 64).

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 parameter, or while executing the command.

RN00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

RN10 (Level 2). Surface does not exist (has not been defined with a SET SURFACE DEFINITIONS command).

RN11 (Level 2). Invalid *priorities* array (there must be an even number of integers in the array, and each integer must be in the range 1 through 4).

RN12 (Level 3). Out of memory while parsing the *priorities* array.

RP11 (Level 2). Invalid number of pixels (must be in the range 0 through 65535).

RP21 (Level 2). There are too many or too few pixels in the code array.

RP22 (Level 3). Parameter 2 memory error (out of memory while parsing the *character-array* parameter.

RQ00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

RQ11 (Level 2). Invalid *view-numbers* array (each view number must range from −2 to 64.)

RQ12 (Level 3). Out of memory while parsing the *view-numbers* array.

RR11 (Level 2). Invalid lower-left coordinates (x must be in the range 0 through 511, and y from 0 through 359).

RR21 (Level 2). Invalid upper-right coordinates (x must be in the range 0 through 511, and y from 0 through 359).

RR31 (Level 2). Invalid fill-index (must be in range 0 through 65535).

RR11 (Level 0). Invalid lower-left coordinates (x value is between range of 480 and 511).

RR21 (Level 0). Invalid upper-right coordinates (x value is between range of 480 and 511).

RS11 (Level 2). Invalid lower-left coordinate (x must be in the range 0 through 511, and y from 0 through 359).

RS21 (Level 2). Invalid upper-right coordinate (x must be in the range 0 through 511, and y from 0 through 359).

RS11 (Level 0). Invalid lower-left coordinate (x value is between range of 480 and 511).

RS21 (Level 0). Invalid upper-right coordinate (x value is between range of 480 and 511).

RU10 (Level 2). Surface does not exist.

RU11 (Level 2). Invalid surface number (must be in the range −1 through 4).

RU21 (Level 2). Invalid ALU mode (must be 0, 7, 11, 12, 15).

RU31 (Level 2). Invalid bits-per-pixel (must be 0, 1, 2, 3, 4, or 6).

RV00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109.)

RV01 (Level 2). Invalid viewport size (the viewport must not be more than 8 times larger than the current window).

RV11 (Level 2). Invalid *first-corner* (x must be in the range 0 through 4095, and y in the range 0 through 3071).

RV21 (Level 2). Invalid *second-corner* (x must be in the range 0 through 4095, and y in the range 0 through 3071).

RX10 (Level 2). The specified *destination-surface* does not exist.

RX11 (Level 2). Invalid destination surface (must be in the range −1 through 4).

RX21 (Level 0). Invalid destination lower-left corner (x must be in range 0 through 639 and y in range 0 through 479).

RX21 (Level 2). Invalid destination lower-left corner (x must be in range 0 through 639 and y in range 0 through 511).

RX31 (Level 0). Invalid first source-corner (x must be in range 0 through 639 and y in range 0 through 479).

RX31 (Level 2). Invalid first source-corner (x must be in range 0 through 639 and y in range 0 through 511).

RX41 (Level 0). Invalid second source-corner (x must be in range 0 through 639 and y in range 0 through 479).

RX41 (Level 2). Invalid second source-corner(x must be in range 0 through 639 and y in range 0 through 511).

SA03 (Level 2). Command invalid at this time; the specified segment is currently being defined.

SA10 (Level 2). Segment does not exist.

SA11 (Level 2). Invalid *segment-number* (must be in the range −3 to −1, or 1 to 32767.)

SA21 (Level 2). Invalid *removal-array*. (Each class number must be −1 or from 1 to 64).

SA22 (Level 3). Out of memory while parsing the *removal-array*.

SA31 (Level 2). Invalid *addition-array* (each class number must be −1 or from 1 to 64).

SA32 (Level 3). Out of memory while parsing the *addition-array*.

SB00 (Level 2). The indicated segment already exists.

SB01 (Level 2). Invalid for next lower segment number (current segment ID is 1).

SB02 (Level 3). Not enough memory to begin segment, or out of memory while ending segment.

SB03 (Level 2). Context error; command is invalid at this time. No segment is currently being defined, or a graphtext character is currently being defined.

SC02 (Level 3). Out of memory while performing command.

SC03 (Level 1). Invalid at this time; no segment is currently being defined.

SD03 (Level 2). Command is invalid at this time (the specified segment is currently being defined.)

SD10 (Level 2). Segment does not exist.

SD11 (Level 2). Invalid *segment-number* (must range from –3 to –1, or 1 to 32767).

SD21 (Level 2). Invalid *detectability* (must be 0 or 1).

SE02 (Level 3). Not enough memory to begin segment, or out of memory while ending segment.

SE03 (Level 2). Command is invalid at this time (a graphtext character or a panel is currently being defined without a segment being defined).

SE10 (Level 2). Segment already exists.

SE11 (Level 2). Invalid *segment-number* (must be in the range 1 through 32767).

SG02 (Level 3). Out of memory while defining font grid.

SG10 (Level 2). Font already exists.

SG11 (Level 2). Invalid *font-number* (must range from 0 to 32767).

SG21 (Level 2). Invalid *grid-width* (must range from 1 to 4095.)

SG31 (Level 2). Invalid *grid-height* (must range from 1 to 4095.)

SH03 (Level 2). Command is invalid at this time (the specified segment is currently being defined.)

SH10 (Level 2). Segment does not exist.

SH11 (Level 2). Invalid *segment-number* (must range from –3 to –1, or from 1 to 32767).

SH21 (Level 2). Invalid *highlighting* (must be 0 or 1).

SI02 (Level 3). Out of memory while transforming segment.

SI03 (Level 2). Command is invalid at this time (the specified segment is currently being defined.)

SI10 (Level 2). Segment does not exist.

SI11 (Level 2). Invalid *segment-number* (must range from –3 to –1, or 1 to 32767).

SI21 (Level 2). Invalid *x-scale-factor* (must range from –32767.0 to 32767.0.)

SI31 (Level 2). Invalid *y-scale-factor* (must range from –32767.0 to 32767.0.)

SI41 (Level 2). Invalid *rotation-angle* (must range from –32767.0 to 32767.0.)

SK02 (Level 3). Out of memory while attempting to delete a segment.

SK10 (Level 1). Segment does not exist.

SK11 (Level 2). Invalid *segment-number* (must be –3, –1, or in the range 1 to 32767).

SL11 (Level 2). Invalid *inclusion-set* (class numbers must be –1, or from 1 to 64).

SL12 (Level 3). Out of memory while parsing the *inclusion-set* array.

SL21 (Level 2). Invalid *exclusion-set* (class numbers must be –1, or from 1 to 64).

SL22 (Level 3). Out of memory while parsing the *exclusion-set* array.

SM03 (Level 2). Command is invalid at this time (the specified segment is currently being defined).

SM10 (Level 2). Segment does not exist.

SM11 (Level 2). Invalid *segment-number* (must range from –3 through 32767).

SM21 (Level 2). Invalid *writing-mode* (must be 0 or 1).

SN00 (Level 2). The indicated segment already exists.

SN01 (Level 2). Invalid for next higher segment number (current segment ID is 32767).

SN02 (Level 3). Out of memory while ending segment.

SN03 (Level 2). No segment is currently being defined, or a graphtext character is currently being defined.

SO02 (Level 3). Out of memory while defining segment.

SO03 (Level 2). Another segment, or a panel is currently being defined.

SO10 (Level 2). Segment already exists.

SO11 (Level 2). Invalid *segment-number* (must range from 1 to 32767).

SQ10 (Level 2). Segment does not exist.

SQ11 (Level 2). Invalid *segment-number* (must range from –3 to 32767).

SQ21 (Level 2). Invalid array of codes. (Must include only the uppercase letters A, D, H, I, M, P, S, V, and X. Also, the array count must be in the range from 0 to 65535.)

SQ22 (Level 3). Out of memory while parsing the parameter.

SR03 (Level 2). Command is invalid at this time (the specified segment is currently being defined).

SR10 (Level 2). Segment does not exist.

SR11 (Level 2). Invalid *old-segment-number* (must range from 1 to 32767).

SR20 (Level 2). Invalid *new-segment-number* (a segment with that number already exists).

SR21 (Level 2). Invalid *new-segment-number* (must range from 1 to 32767).

SS03 (Level 2). Command is invalid at this time (the specified segment is currently being defined).

SS10 (Level 2). Segment does not exist.

SS11 (Level 2). Invalid *segment-number* (must range from –3 to –1, or from 1 to 32767).

SS21 (Level 2). Invalid *priority-number* (must range from –32768 to 32767).

ST02 (Level 3). Out of memory while defining graphtext character.

ST03 (Level 2). Command is invalid at this time (a graphtext character is currently being defined).

ST10 (Level 2). The *font* specified has no grid defined for it.

ST11 (Level 2). Invalid *font* number (must be in the range 0 through 32767).

ST20 (Level 2). The *character* specified has already been defined in this *font*.

ST21 (Level 2). Invalid *character* number (must be in the range 32 to 126).

SU03 (Level 1). Command is invalid at this time (no graphtext character is being defined).

SV02 (Level 3). Out of memory while performing command.

SV03 (Level 2). Command is invalid at this time (the specified segment is currently being defined).

SV10 (Level 2). Segment does not exist.

SV11 (Level 2). Invalid *segment-number* (must range from –3 to 32767).

SV21 (Level 2). Invalid *visibility* (must be 0 or 1).

SX02 (Level 3). Out of memory while performing command.

SX03 (Level 2). Command is invalid at this time (the specified segment is currently being defined).

SX10 (Level 2). Segment does not exist.

SX11 (Level 2). Invalid *segment-number* (must range from –3 to 32767).

SZ03 (Level 2). Command is invalid at this time (a graphtext character is currently being defined).

SZ10 (Level 1). The specified font does not exist (no grid has been defined for that font).

SZ11 (Level 2). Invalid *font-number* (must range from –1 to 32767).

SZ20 (Level 1). The character specified does not exist in this font.

SZ21 (Level 2). Invalid *char-number* (must be –1, or in the range 32 to 126).

TB00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

TB11 (Level 2). Invalid *first-color-coordinate* (if in HLS mode, must be in the range –32768 through 32767; if in RGB or CMY mode, must be in the range 0 through 100).

TB21 (Level 2). Invalid *second-color-coordinate* (must be in the range 0 through 100).

TB31 (Level 2). Invalid *third-color-coordinate* (must be in the range 0 through 100).

TC11 (Level 2). Invalid hue value (must be between –32768 and 32767)

TC21 (Level 2). Invalid lightness value (must be between 0 and 100)

TC31 (Level 2). Invalid saturation value (must be between 0 and 100)

TD11 (Level 2). Invalid index value (must be between 0 and 65535)

TD21 (Level 2). Invalid index value (must be between 0 and 65535)

TF11 (Level 2). Invalid color-mixtures array (must be an array of quadruples consisting of a color index and the three HLS color coordinates).

TF12 (Level 3). Parameter 2 memory error (out of memory while parsing the color-mixtures array).

TG11 (Level 2). Invalid surface number (must be –1, or in the range 1 through 4).

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). Parameter 2 memory error (out of memory while parsing the color-mixtures array).

TM00 (Level 0). Unrecognized command (terminal is not a 4107 or 4109).

TM11 (Level 2). Invalid *color-specifying-mode* (must range from 0 to 3).

TM21 (Level 2). Invalid *color-overlay-mode* (must range from 0 to 3).

TM31 (Level 2). Invalid gray-mode (must be 0 or 1).

[A11 (Level 2). Invalid value (must be in the range of 0 to 32767).

[B11 (Level 2). Invalid value (must be in the range of 0 to 32767).

[C11 (Level 2). Invalid value (must be in the range of 0 to 32767).

[c11 (Level 2). Invalid first parameter (must be 0 or omitted).

[D11 (Level 2). Invalid value (must be in the range of 0 to 32767).

[f11 (Level 2). Invalid value for parameter 1 (must be in the range of 0 to 32767).

[f21 (Level 2). Invalid value for parameter 2 (must be in the range of 0 to 32767).

[g11 (Level 2). Invalid value (must be 0, 2 or 3).

[h01 (Level 2). Invalid mode value (must be either 4, 20, <1, ?1, ?3, ?6, ?7, or ?8).

[h10 to **[h90 (Level 2).** Invalid mode value.

[h11 to **[h91 (Level 2).** Invalid parameter syntax.

[H11 (Level 2). Invalid value for parameter 1 (must be in the range of 0 to 32767).

[H21 (Level 2). Invalid value for parameter 2 (must be in the range of 0 to 32767).

[I11 (Level 2). Invalid value (must be in the range of 0 to 32767).

[J11 (Level 2). Invalid value (must be 0, 1 or 2).

[K11 (Level 2). Invalid value (must be 0, 1 or 2).

[l01 (Level 2). Invalid mode value (must be either 4, 20, <1, ?1, ?3, ?6, ?7, or ?8).

[l10 to **[l90 (Level 0).** Unrecognized mode value (treated as a no-op).

[l11 to **[l91 (Level 2).** Invalid parameter syntax.

[L11 (Level 2). Invalid value (must be in the range of 0 to 32767).

[m01 (Level 2). Invalid rendition value (must be either 0, 1, 4, 5, 7, <, >, or =).

[M11 (Level 2). Invalid value (must be in the range of 0 to 32767).

[n11 (Level 2). Invalid value for parameter 1 (must be 6).

[P11 (Level 2). Invalid value (must be in the range of 0 to 32767).

[r11 (Level 2). Invalid value for parameter 1 (must be in the range of 0 to 32767).

[r21 (Level 2). Invalid value for parameter 2 (must be in the range of 0 to 32767).

[S11 (Level 2). Invalid value (must be in the range of 0 to 32767).

[T11 (Level 2). Invalid value (must be in the range of 0 to 32767).

[X11 (Level 2). Invalid value (must be in the range of 0 to 32767).

[Z11 (Level 2). Invalid value (must be in the range of 0 to 32767).

[@11 (Level 2). Invalid value (must be in the range of 0 to 32767).

%!11 (Level 2). Invalid parameter R(must be 0, 1, or 2).

#!11 (Level 2). Invalid parameter (must be 0).

SPA11 (Level 2). Invalid value (must be 0 to 32767).

SP@11 (Level 2). Invalid value (must be 0 to 32767).

Appendix C

COMMAND SUMMARY

BEGIN GRAPHTEXT CHARACTER
BEGIN HIGHER SEGMENT
BEGIN LOWER SEGMENT
BEGIN NEW SEGMENT
BEGIN PANEL BOUNDARY
BEGIN PIXEL OPERATIONS
BEGIN SEGMENT
CANCEL
CLEAR DIALOG SCROLL
COPY

CRLF
DEFINE MACRO
DELETE GRAPHTEXT CHARACTER
DELETE SEGMENT
DELETE VIEW
DISABLE GIN
DRAW
DRAW MARKER
ENABLE DIALOG AREA
ENABLE GIN

ENABLE 4010 GIN
END GRAPHTEXT CHARACTER
END PANEL
END SEGMENT
ENTER ALPHA MODE
ENTER BYPASS MODE
ENTER MARKER MODE
ENTER VECTOR MODE
EXPAND MACRO
FACTORY

GRAPHIC TEXT
HARDCOPY
HELP
IGNORE DELETES
INCLUDE COPY OF SEGMENT
LEARN/NVLEARN
LFCR
LOCAL
LOCK KEYBOARD
LOCK VIEWING KEYS

MACRO STATUS
MAP INDEX TO PEN
MOVE
PAGE
PIXEL COPY
PLOT
PORT ASSIGN
PROMPT MODE
RASTER WRITE
RECTANGLE FILL

RENAME SEGMENT
RENEW VIEW
REPORT ERRORS
REPORT SYNTAX MODE
REPORT DEVICE STATUS
REPORT GIN POINT
REPORT PORT STATUS
REPORT SEGMENT STATUS
REPORT TERMINAL SETTINGS
REPORT 4010 STATUS

RESET
RUNLENGTH WRITE
SAVE NONVOLATILE PARAMETERS
SELECT CODE
SELECT FILL PATTERN
SELECT HARDCOPY INTERFACE
SELECT VIEW
SET ALPHA CURSOR INDICES
SET ALPHATEXT FONT
SET BACKGROUND COLOR

SET BACKGROUND INDICES
SET BAUD RATES
SET BORDER VISIBILITY
SET BREAK TIME
SET BYPASS CANCEL CHARACTER
SET CHARACTER PATH
SET COLOR MODE
SET COPY SIZE
SET CURRENT MATCHING CLASS
SET DIALOG AREA BUFFER SIZE

SET DIALOG AREA COLOR MAP
SET DIALOG HARDCOPY ATTRIBUTES
SET DIALOG AREA INDEX
SET DIALOG AREA LINES
SET DIALOG AREA VISIBILITY
SET DIALOG AREA WRITING MODE
SET ECHO
SET EDIT CHARACTERS
SET EOF STRING
SET EOL STRING

SET EOM CHARACTERS
SET ERROR THRESHOLD
SET FIXUP LEVEL
SET FLAGGING MODE
SET GIN AREA
SET GIN CURSOR
SET GIN CURSOR COLOR
SET GIN DISPLAY START POINT
SET GIN GRIDDING
SET GIN INKING

SET GIN RUBBERBANDING
SET GIN STROKE FILTERING
SET GIN WINDOW
SET GRAPHICS AREA WRITING MODE
SET GRAPHTEXT FONT
SET GRAPHTEXT FONT GRID
SET GRAPHTEXT PRECISION
SET GRAPHTEXT ROTATION
SET GRAPHTEXT SIZE
SET GRAPHTEXT SLANT

SET KEY EXECUTE CHARACTER
SET LINE INDEX
SET LINE STYLE
SET MARKER TYPE
SET PARITY
SET PICK APERTURE
SET PICK ID
SET PIVOT POINT
SET PIXEL BEAM POSITION
SET PIXEL VIEWPORT

SET PORT BAUD RATE
SET PORT EOF STRING
SET PORT FLAGGING MODE
SET PORT PARITY
SET PORT STOP BITS
SET PROMPT STRING
SET QUEUE SIZE
SET REPORT EOM FREQUENCY
SET REPORT MAX LINE LENGTH
SET REPORT SIG CHARACTERS

SET SEGMENT CLASS
SET SEGMENT DETECTABILITY
SET SEGMENT DISPLAY PRIORITY
SET SEGMENT HIGHLIGHTING
SET SEGMENT IMAGE TRANSFORM
SET SEGMENT POSITION
SET SEGMENT VISIBILITY
SET SEGMENT WRITING MODE
SET SNOOPY MODE
SET STOP BITS

SET SURFACE COLOR MAP
SET SURFACE DEFINITIONS
SET SURFACE PRIORITIES
SET SURFACE VISIBILITY
SET TAB STOPS
SET TEXT INDEX
SET TRANSMIT DELAY
SET TRANSMIT RATE LIMIT
SET VIEW ATTRIBUTES
SET VIEW DISPLAY CLUSTER

SET VIEWPORT
SET WINDOW
SET 4014 ALPHATEXT SIZE
SET 4014 LINE STYLE
STATUS
4010 HARDCOPY

Appendix D

EXAMPLES OF INTEGER PARAMETERS

Table D-1 lists integer parameters between −1049 and +1049.

Table D-1

INTEGER PARAMETERS

+ Int[a]	Param	−Int[b]	Param
0	0	−0	^{S}P
1	1	−1	!
2	2	−2	”
3	3	−3	#
4	4	−4	$
5	5	−5	%
6	6	−6	&
7	7	−7	’
8	8	−8	(
9	9	−9	)
10	:	−10	*
11	;	−11	+
12	<	−12	,
13	=	−13	−
14	>	−14	.
15	?	−15	/
16	A0	−16	A^{S}P
17	A1	−17	A!
18	A2	−18	A”
19	A3	−19	A#
20	A4	−20	A$
21	A5	−21	A%
22	A6	−22	A&
23	A7	−23	A’
24	A8	−24	A(
25	A9	−25	A)
26	A:	−26	A*
27	A;	−27	A+
28	A<	−28	A,
29	A=	−29	A−
30	A>	−30	A.
31	A?	−31	A/
32	B0	−32	B^{S}P
33	B1	−33	B!
34	B2	−34	B”
35	B3	−35	B#
36	B4	−36	B$
37	B5	−37	B%
38	B6	−38	B&
39	B7	−39	B’
40	B8	−40	B(
41	B9	−41	B)
42	B:	−42	B*
43	B;	−43	B+
44	B<	−44	B,
45	B=	−45	B−
46	B>	−46	B.
47	B?	−47	B/
48	C0	−48	C^{S}P
49	C1	−49	C!
50	C2	−50	C”
51	C3	−51	C#
52	C4	−52	C$
53	C5	−53	C%
54	C6	−54	C&
55	C7	−55	C’
56	C8	−56	C(
57	C9	−57	C)
58	C:	−58	C*
59	C;	−59	C+
60	C<	−60	C,
61	C=	−61	C−
62	C>	−62	C.
63	C?	−63	C/
64	D0	−64	D^{S}P
65	D1	−65	D!
66	D2	−66	D”
67	D3	−67	D#
68	D4	−68	D$
69	D5	−69	D%
70	D6	−70	D&
71	D7	−71	D’
72	D8	−72	D(
73	D9	−73	D)
74	D:	−74	D*
75	D;	−75	D+
76	D<	−76	D,
77	D=	−77	D−
78	D>	−78	D.
79	D?	−79	D/
80	E0	−80	E^{S}P
81	E1	−81	E!
82	E2	−82	E”
83	E3	−83	E#
84	E4	−84	E$
85	E5	−85	E%
86	E6	−86	E&
87	E7	−87	E’
88	E8	−88	E(
89	E9	−89	E)
90	E:	−90	E*
91	E;	−91	E+
92	E<	−92	E,
93	E=	−93	E−
94	E>	−94	E.
95	E?	−95	E/
96	F0	−96	F^{S}P
97	F1	−97	F!
98	F2	−98	F”
99	F3	−99	F#
100	F4	−100	F$
101	F5	−101	F%
102	F6	−102	F&
103	F7	−103	F’
104	F8	−104	F(
105	F9	−105	F)
106	F:	−106	F*
107	F;	−107	F+
108	F<	−108	F,
109	F=	−109	F−
110	F>	−110	F.
111	F?	−111	F/
112	G0	−112	G^{S}P
113	G1	−113	G!
114	G2	−114	G”
115	G3	−115	G#
116	G4	−116	G$
117	G5	−117	G%
118	G6	−118	G&
119	G7	−119	G’
120	G8	−120	G(
121	G9	−121	G)
122	G:	−122	G*
123	G;	−123	G+
124	G<	−124	G,
125	G=	−125	G−
126	G>	−126	G.
127	G?	−127	G/
128	H0	−128	H^{S}P
129	H1	−129	H!
130	H2	−130	H”
131	H3	−131	H#
132	H4	−132	H$
133	H5	−133	H%
134	H6	−134	H&
135	H7	−135	H’
136	H8	−136	H(
137	H9	−137	H)
138	H:	−138	H*
139	H;	−139	H+
140	H<	−140	H,
141	H=	−141	H−
142	H>	−142	H.
143	H?	−143	H/
144	I0	−144	I^{S}P
145	I1	−145	I!
146	I2	−146	I”
147	I3	−147	I#
148	I4	−148	I$
149	I5	−149	I%

[a] Positive integer

[b] Negative integer

For integers ± 144 through ± 159, the first character of the parameter is an uppercase I (ADE 73).

Table D-1 (cont)

INTEGER PARAMETERS

+Int[a]	Param	−Int[b]	Param
150	I6	−150	I&
151	I7	−151	I’
152	I8	−152	I(
153	I9	−153	I)
154	I:	−154	I*
155	I;	−155	I+
156	I<	−156	I,
157	I=	−157	I−
158	I>	−158	I.
159	I?	−159	I/
160	J0	−160	J^{SP}
161	J1	−161	J!
162	J2	−162	J”
163	J3	−163	J#
164	J4	−164	J$
165	J5	−165	J%
166	J6	−166	J&
167	J7	−167	J’
168	J8	−168	J(
169	J9	−169	J)
170	J:	−170	J*
171	J;	−171	J+
172	J<	−172	J,
173	J=	−173	J−
174	J>	−174	J.
175	J?	−175	J/
176	K0	−176	K^{SP}
177	K1	−177	K!
178	K2	−178	K”
179	K3	−179	K#
180	K4	−180	K$
181	K5	−181	K%
182	K6	−182	K&
183	K7	−183	K’
184	K8	−184	K(
185	K9	−185	K)
186	K:	−186	K*
187	K;	−187	K+
188	K<	−188	K,
189	K=	−189	K−
190	K>	−190	K.
191	K?	−191	K/
192	L0	−192	L^{SP}
193	L1	−193	L!
194	L2	−194	L”
195	L3	−195	L#
196	L4	−196	L$
197	L5	−197	L%
198	L6	−198	L&
199	L7	−199	L’
200	L8	−200	L(
201	L9	−201	L)
202	L:	−202	L*
203	L;	−203	L+
204	L<	−204	L,
205	L=	−205	L−
206	L>	−206	L.
207	L?	−207	L/
208	M0	−208	M^{SP}
209	M1	−209	M!
210	M2	−210	M”
211	M3	−211	M#
212	M4	−212	M$
213	M5	−213	M%
214	M6	−214	M&
215	M7	−215	M’
216	M8	−216	M(
217	M9	−217	M)
218	M:	−218	M*
219	M;	−219	M+
220	M<	−220	M,
221	M=	−221	M−
222	M>	−222	M.
223	M?	−223	M/
224	N0	−224	N^{SP}
225	N1	−225	N!
226	N2	−226	N”
227	N3	−227	N#
228	N4	−228	N$
229	N5	−229	N%
230	N6	−230	N&
231	N7	−231	N’
232	N8	−232	N(
233	N9	−233	N)
234	N:	−234	N*
235	N;	−235	N+
236	N<	−236	N,
237	N=	−237	N−
238	N>	−238	N.
239	N?	−239	N/
240	O0	−240	O^{SP}
241	O1	−241	O!
242	O2	−242	O”
243	O3	−243	O#
244	O4	−244	O$
245	O5	−245	O%
246	O6	−246	O&
247	O7	−247	O’
248	O8	−248	O(
249	O9	−249	O)
250	O:	−250	O*
251	O;	−251	O+
252	O<	−252	O,
253	O=	−253	O−
254	O>	−254	O.
255	O?	−255	O/
256	P0	−256	P^{SP}
257	P1	−257	P!
258	P2	−258	P”
259	P3	−259	P#
260	P4	−260	P$
261	P5	−261	P%
262	P6	−262	P&
263	P7	−263	P’
264	P8	−264	P(
265	P9	−265	P)
266	P:	−266	P*
267	P;	−267	P+
268	P<	−268	P,
269	P=	−269	P−
270	P>	−270	P.
271	P?	−271	P/
272	Q0	−272	Q^{SP}
273	Q1	−273	Q!
274	Q2	−274	Q”
275	Q3	−275	Q#
276	Q4	−276	Q$
277	Q5	−277	Q%
278	Q6	−278	Q&
279	Q7	−279	Q’
280	Q8	−280	Q(
281	Q9	−281	Q)
282	Q:	−282	Q*
283	Q;	−283	Q+
284	Q<	−284	Q,
285	Q=	−285	Q−
286	Q>	−286	Q.
287	Q?	−287	Q/
288	R0	−288	R^{SP}
289	R1	−289	R!
290	R2	−290	R”
291	R3	−291	R#
292	R4	−292	R$
293	R5	−293	R%
294	R6	−294	R&
295	R7	−295	R’
296	R8	−296	R(
297	R9	−297	R)
298	R:	−298	R*
299	R;	−299	R+

[a] Positive integer

[b] Negative integer

For integers ± 144 through ± 159, the first character of the parameter is an uppercase I (ADE 73).

Table D-1 (cont)

INTEGER PARAMETERS

+ Int[a]	Param	-Int[b]	Param
300	R<	−300	R,
301	R=	−301	R−
302	R>	−302	R.
303	R?	−303	R/
304	S0	−304	S$^{S}_{P}$
305	S1	−305	S!
306	S2	−306	S"
307	S3	−307	S#
308	S4	−308	S$
309	S5	−309	S%
310	S6	−310	S&
311	S7	−311	S'
312	S8	−312	S(
313	S9	−313	S)
314	S:	−314	S*
315	S;	−315	S+
316	S<	−316	S,
317	S=	−317	S−
318	S>	−318	S.
319	S?	−319	S/
320	T0	−320	T$^{S}_{P}$
321	T1	−321	T!
322	T2	−322	T"
323	T3	−323	T#
324	T4	−324	T$
325	T5	−325	T%
326	T6	−326	T&
327	T7	−327	T'
328	T8	−328	T(
329	T9	−329	T)
330	T:	−330	T*
331	T;	−331	T+
332	T<	−332	T,
333	T=	−333	T−
334	T>	−334	T.
335	T?	−335	T/
336	U0	−336	U$^{S}_{P}$
337	U1	−337	U!
338	U2	−338	U"
339	U3	−339	U#
340	U4	−340	U$
341	U5	−341	U%
342	U6	−342	U&
343	U7	−343	U'
344	U8	−344	U(
345	U9	−345	U)
346	U:	−346	U*
347	U;	−347	U+
348	U<	−348	U,
349	U=	−349	U−
350	U>	−350	U.
351	U?	−351	U/
352	V0	−352	V$^{S}_{P}$
353	V1	−353	V!
354	V2	−354	V"
355	V3	−355	V#
356	V4	−356	V$
357	V5	−357	V%
358	V6	−358	V&
359	V7	−359	V'
360	V8	−360	V(
361	V9	−361	V)
362	V:	−362	V*
363	V;	−363	V+
364	V<	−364	V,
365	V=	−365	V−
366	V>	−366	V.
367	V?	−367	V/
368	W0	−368	W$^{S}_{P}$
369	W1	−369	W!
370	W2	−370	W"
371	W3	−371	W#
372	W4	−372	W$
373	W5	−373	W%
374	W6	−374	W&
375	W7	−375	W'
376	W8	−376	W(
377	W9	−377	W)
378	W:	−378	W*
379	W;	−379	W+
380	W<	−380	W,
381	W=	−381	W−
382	W>	−382	W.
383	W?	−383	W/
384	X0	−384	X$^{S}_{P}$
385	X1	−385	X!
386	X2	−386	X"
387	X3	−387	X#
388	X4	−388	X$
389	X5	−389	X%
390	X6	−390	X&
391	X7	−391	X'
392	X8	−392	X(
393	X9	−393	X)
394	X:	−394	X*
395	X;	−395	X+
396	X<	−396	X,
397	X=	−397	X−
398	X>	−398	X.
399	X?	−399	X/
400	Y0	−400	Y$^{S}_{P}$
401	Y1	−401	Y!
402	Y2	−402	Y"
403	Y3	−403	Y#
404	Y4	−404	Y$
405	Y5	−405	Y%
406	Y6	−406	Y&
407	Y7	−407	Y'
408	Y8	−408	Y(
409	Y9	−409	Y)
410	Y:	−410	Y*
411	Y;	−411	Y+
412	Y<	−412	Y,
413	Y=	−413	Y−
414	Y>	−414	Y.
415	Y?	−415	Y/
416	Z0	−416	Z$^{S}_{P}$
417	Z1	−417	Z!
418	Z2	−418	Z"
419	Z3	−419	Z#
420	Z4	−420	Z$
421	Z5	−421	Z%
422	Z6	−422	Z&
423	Z7	−423	Z'
424	Z8	−424	Z(
425	Z9	−425	Z)
426	Z:	−426	Z*
427	Z;	−427	Z+
428	Z<	−428	Z,
429	Z=	−429	Z−
430	Z>	−430	Z.
431	Z?	−431	Z/
432	[0	−432	[$^{S}_{P}$
433	[1	−433	[!
434	[2	−434	["
435	[3	−435	[#
436	[4	−436	[$
437	[5	−437	[%
438	[6	−438	[&
439	[7	−439	['
440	[8	−440	[(
441	[9	−441	[)
442	[:	−442	[*
443	[;	−443	[+
444	[<	−444	[,
445	[=	−445	[−
446	[>	−446	[.
447	[?	−447	[/
448	\0	−448	\$^{S}_{P}$
449	\1	−449	\!

[a] Positive integer

[b] Negative integer

Table D-1 (cont)

INTEGER PARAMETERS

+Int[a]	Param	-Int[b]	Param	+Int[a]	Param	-Int[b]	Param	+Int[a]	Param	-Int[b]	Param
450	\2	−450	\”	500	_4	−500	_$	550	b6	−550	b&
451	\3	−451	\#	501	_5	−501	_%	551	b7	−551	b’
452	\4	−452	\$	502	_6	−502	_&	552	b8	−552	b(
453	\5	−453	\%	503	_7	−503	_’	553	b9	−553	b)
454	\6	−454	\&	504	_8	−504	_(	554	b:	−554	b*
455	\7	−455	\’	505	_9	−505	_)	555	b;	−555	b+
456	\8	−456	\(	506	_:	−506	_*	556	b<	−556	b,
457	\9	−457	\)	507	_;	−507	_+	557	b=	−557	b−
458	\:	−458	*	508	_<	−508	_,	558	b>	−558	b.
459	\;	−459	\+	509	_=	−509	_−	559	b?	−559	b/
460	\<	−460	\,	510	_>	−510	_.	560	c0	−560	c^{S_P}
461	\=	−461	\−	511	_?	−511	_/	561	c1	−561	c!
462	\>	−462	\.	512	‘0	−512	‘^{S_P}	562	c2	−562	c”
463	\?	−463	\/	513	‘1	−513	‘!	563	c3	−563	c#
464	]0	−464	]^{S_P}	514	‘2	−514	‘”	564	c4	−564	c$
465	]1	−465	]!	515	‘3	−515	‘#	565	c5	−565	c%
466	]2	−466	]”	516	‘4	−516	‘$	566	c6	−566	c&
467	]3	−467	]#	517	‘5	−517	‘%	567	c7	−567	c’
468	]4	−468	]$	518	‘6	−518	‘&	568	c8	−568	c(
469	]5	−469	]%	519	‘7	−519	‘’	569	c9	−569	c)
470	]6	−470	]&	520	‘8	−520	‘(	570	c:	−570	c*
471	]7	−471	]’	521	‘9	−521	‘)	571	c;	−571	c+
472	]8	−472	](	522	‘:	−522	‘*	572	c<	−572	c,
473	]9	−473	])	523	‘;	−523	‘+	573	c=	−573	c−
474	]:	−474	]*	524	‘<	−524	‘,	574	c>	−574	c.
475	];	−475	]+	525	‘=	−525	‘−	575	c?	−575	c/
476	]<	−476	],	526	‘>	−526	‘.	576	d0	−576	d^{S_P}
477	]=	−477	]−	527	‘?	−527	‘/	577	d1	−577	d!
478	]>	−478	].	528	a0	−528	a^{S_P}	578	d2	−578	d”
479	]?	−479	]/	529	a1	−529	a!	579	d3	−579	d#
480	∧0	−480	∧^{S_P}	530	a2	−530	a”	580	d4	−580	d$
481	∧1	−481	∧!	531	a3	−531	a#	581	d5	−581	d%
482	∧2	−482	∧”	532	a4	−532	a$	582	d6	−582	d&
483	∧3	−483	∧#	533	a5	−533	a%	583	d7	−583	d’
484	∧4	−484	∧$	534	a6	−534	a&	584	d8	−584	d(
485	∧5	−485	∧%	535	a7	−535	a’	585	d9	−585	d)
486	∧6	−486	∧&	536	a8	−536	a(	586	d:	−586	d*
487	∧7	−487	∧’	537	a9	−537	a)	587	d;	−587	d+
488	∧8	−488	∧(	538	a:	−538	a*	588	d<	−588	d,
489	∧9	−489	∧)	539	a;	−539	a+	589	d=	−589	d−
490	∧:	−490	∧*	540	a<	−540	a,	590	d>	−590	d.
491	∧;	−491	∧+	541	a=	−541	a−	591	d?	−591	d/
492	∧<	−492	∧,	542	a>	−542	a.	592	e0	−592	e^{S_P}
493	∧=	−493	∧−	543	a?	−543	a/	593	e1	−593	e!
494	∧>	−494	∧.	544	b0	−544	b^{S_P}	594	e2	−594	e”
495	∧?	−495	∧/	545	b1	−545	b!	595	e3	−595	e#
496	_0	−496	_^{S_P}	546	b2	−546	b”	596	e4	−596	e$
497	_1	−497	_!	547	b3	−547	b#	597	e5	−597	e%
498	_2	−498	_”	548	b4	−548	b$	598	e6	−598	e&
499	_3	−499	_#	549	b5	−549	b%	599	e7	−599	e’

[a] Positive integer

[b] Negative integer

Table D-1 (cont)

INTEGER PARAMETERS

+ Int[a]	Param	-Int[b]	Param
600	e8	−600	e(
601	e9	−601	e)
602	e:	−602	e*
603	e;	−603	e+
604	e<	−604	e,
605	e=	−605	e−
606	e>	−606	e.
607	e?	−607	e/
608	f0	−608	f^{SP}
609	f1	−609	f!
610	f2	−610	f”
611	f3	−611	f#
612	f4	−612	f$
613	f5	−613	f%
614	f6	−614	f&
615	f7	−615	f’
616	f8	−616	f(
617	f9	−617	f)
618	f:	−618	f*
619	f;	−619	f+
620	f<	−620	f,
621	f=	−621	f−
622	f>	−622	f.
623	f?	−623	f/
624	g0	−624	g^{SP}
625	g1	−625	g!
626	g2	−626	g”
627	g3	−627	g#
628	g4	−628	g$
629	g5	−629	g%
630	g6	−630	g&
631	g7	−631	g’
632	g8	−632	g(
633	g9	−633	g)
634	g:	−634	g*
635	g;	−635	g+
636	g<	−636	g,
637	g=	−637	g−
638	g>	−638	g.
639	g?	−639	g/
640	h0	−640	h^{SP}
641	h1	−641	h!
642	h2	−642	h”
643	h3	−643	h#
644	h4	−644	h$
645	h5	−645	h%
646	h6	−646	h&
647	h7	−647	h’
648	h8	−648	h(
649	h9	−649	h)
650	h:	−650	h*
651	h;	−651	h+
652	h<	−652	h,
653	h=	−653	h−
654	h>	−654	h.
655	h?	−655	h/
656	i0	−656	i^{SP}
657	i1	−657	i!
658	i2	−658	i”
659	i3	−659	i#
660	i4	−660	i$
661	i5	−661	i%
662	i6	−662	i&
663	i7	−663	i’
664	i8	−664	i(
665	i9	−665	i)
666	i:	−666	i*
667	i;	−667	i+
668	i<	−668	i,
669	i=	−669	i−
670	i>	−670	i.
671	i?	−671	i/
672	j0	−672	j^{SP}
673	j1	−673	j!
674	j2	−674	j”
675	j3	−675	j#
676	j4	−676	j$
677	j5	−677	j%
678	j6	−678	j&
679	j7	−679	j’
680	j8	−680	j(
681	j9	−681	j)
682	j:	−682	j*
683	j;	−683	j+
684	j<	−684	j,
685	j=	−685	j−
686	j>	−686	j.
687	j?	−687	j/
688	k0	−688	k^{SP}
689	k1	−689	k!
690	k2	−690	k”
691	k3	−691	k#
692	k4	−692	k$
693	k5	−693	k%
694	k6	−694	k&
695	k7	−695	k’
696	k8	−696	k(
697	k9	−697	k)
698	k:	−698	k*
699	k;	−699	k+
700	k<	−700	k,
701	k=	−701	k−
702	k>	−702	k.
703	k?	−703	k/
704	l0	−704	l^{SP}
705	l1	−705	l!
706	l2	−706	l”
707	l3	−707	l#
708	l4	−708	l$
709	l5	−709	l%
710	l6	−710	l&
711	l7	−711	l’
712	l8	−712	l(
713	l9	−713	l)
714	l:	−714	l*
715	l;	−715	l+
716	l<	−716	l,
717	l=	−717	l−
718	l>	−718	l.
719	l?	−719	l/
720	m0	−720	m^{SP}
721	m1	−721	m!
722	m2	−722	m”
723	m3	−723	m#
724	m4	−724	m$
725	m5	−725	m%
726	m6	−726	m&
727	m7	−727	m’
728	m8	−728	m(
729	m9	−729	m)
730	m:	−730	m*
731	m;	−731	m+
732	m<	−732	m,
733	m=	−733	m−
734	m>	−734	m.
735	m?	−735	m/
736	n0	−736	n^{SP}
737	n1	−737	n!
738	n2	−738	n”
739	n3	−739	n#
740	n4	−740	n$
741	n5	−741	n%
742	n6	−742	n&
743	n7	−743	n’
744	n8	−744	n(
745	n9	−745	n)
746	n:	−746	n*
747	n;	−747	n+
748	n<	−748	n,
749	n=	−749	n−

[a] Positive integer

[b] Negative integer

For integers ± 704 through ± 719, the first character of the parameter is a lowercase l (ADE 108).

Table D-1 (cont)

INTEGER PARAMETERS

+Int[a]	Param	-Int[b]	Param
750	n>	−750	n.
751	n?	−751	n/
752	o0	−752	o$^{S}_{P}$
753	o1	−753	o!
754	o2	−754	o"
755	o3	−755	o#
756	o4	−756	o$
757	o5	−757	o%
758	o6	−758	o&
759	o7	−759	o'
760	o8	−760	o(
761	o9	−761	o)
762	o:	−762	o*
763	o;	−763	o+
764	o<	−764	o,
765	o=	−765	o−
766	o>	−766	o.
767	o?	−767	o/
768	p0	−768	p$^{S}_{P}$
769	p1	−769	p!
770	p2	−770	p"
771	p3	−771	p#
772	p4	−772	p$
773	p5	−773	p%
774	p6	−774	p&
775	p7	−775	p'
776	p8	−776	p(
777	p9	−777	p)
778	p:	−778	p*
779	p;	−779	p+
780	p<	−780	p,
781	p=	−781	p−
782	p>	−782	p.
783	p?	−783	p/
784	q0	−784	q$^{S}_{P}$
785	q1	−785	q!
786	q2	−786	q"
787	q3	−787	q#
788	q4	−788	q$
789	q5	−789	q%
790	q6	−790	q&
791	q7	−791	q'
792	q8	−792	q(
793	q9	−793	q)
794	q:	−794	q*
795	q;	−795	q+
796	q<	−796	q,
797	q=	−797	q−
798	q>	−798	q.
799	q?	−799	q/
800	r0	−800	r$^{S}_{P}$
801	r1	−801	r!
802	r2	−802	r"
803	r3	−803	r#
804	r4	−804	r$
805	r5	−805	r%
806	r6	−806	r&
807	r7	−807	r'
808	r8	−808	r(
809	r9	−809	r)
810	r:	−810	r*
811	r;	−811	r+
812	r<	−812	r,
813	r=	−813	r−
814	r>	−814	r.
815	r?	−815	r/
816	s0	−816	s$^{S}_{P}$
817	s1	−817	s!
818	s2	−818	s"
819	s3	−819	s#
820	s4	−820	s$
821	s5	−821	s%
822	s6	−822	s&
823	s7	−823	s'
824	s8	−824	s(
825	s9	−825	s)
826	s:	−826	s*
827	s;	−827	s+
828	s<	−828	s,
829	s=	−829	s−
830	s>	−830	s.
831	s?	−831	s/
832	t0	−832	t$^{S}_{P}$
833	t1	−833	t!
834	t2	−834	t"
835	t3	−835	t#
836	t4	−836	t$
837	t5	−837	t%
838	t6	−838	t&
839	t7	−839	t'
840	t8	−840	t(
841	t9	−841	t)
842	t:	−842	t*
843	t;	−843	t+
844	t<	−844	t,
845	t=	−845	t−
846	t>	−846	t.
847	t?	−847	t/
848	u0	−848	u$^{S}_{P}$
849	u1	−849	u!
850	u2	−850	u"
851	u3	−851	u#
852	u4	−852	u$
853	u5	−853	u%
854	u6	−854	u&
855	u7	−855	u'
856	u8	−856	u(
857	u9	−857	u)
858	u:	−858	u*
859	u;	−859	u+
860	u<	−860	u,
861	u=	−861	u−
862	u>	−862	u.
863	u?	−863	u/
864	v0	−864	v$^{S}_{P}$
865	v1	−865	v!
866	v2	−866	v"
867	v3	−867	v#
868	v4	−868	v$
869	v5	−869	v%
870	v6	−870	v&
871	v7	−871	v'
872	v8	−872	v(
873	v9	−873	v)
874	v:	−874	v*
875	v;	−875	v+
876	v<	−876	v,
877	v=	−877	v−
878	v>	−878	v.
879	v?	−879	v/
880	w0	−880	w$^{S}_{P}$
881	w1	−881	w!
882	w2	−882	w"
883	w3	−883	w#
884	w4	−884	w$
885	w5	−885	w%
886	w6	−886	w&
887	w7	−887	w'
888	w8	−888	w(
889	w9	−889	w)
890	w:	−890	w*
891	w;	−891	w+
892	w<	−892	w,
893	w=	−893	w−
894	w>	−894	w.
895	w?	−895	w/
896	x0	−896	x$^{S}_{P}$
897	x1	−897	x!
898	x2	−898	x"
899	x3	−899	x#

[a] Positive integer

[b] Negative integer

Table D-1 (cont)

INTEGER PARAMETERS

+Int[a]	Param	-Int[b]	Param
900	x4	-900	x$
901	x5	-901	x%
902	x6	-902	x&
903	x7	-903	x’
904	x8	-904	x(
905	x9	-905	x)
906	x:	-906	x*
907	x;	-907	x+
908	x<	-908	x,
909	x=	-909	x-
910	x>	-910	x.
911	x?	-911	x/
912	y0	-912	y$^{S}_{P}$
913	y1	-913	y!
914	y2	-914	y”
915	y3	-915	y#
916	y4	-916	y$
917	y5	-917	y%
918	y6	-918	y&
919	y7	-919	y’
920	y8	-920	y(
921	y9	-921	y)
922	y:	-922	y*
923	y;	-923	y+
924	y<	-924	y,
925	y=	-925	y-
926	y>	-926	y.
927	y?	-927	y/
928	z0	-928	z$^{S}_{P}$
929	z1	-929	z!
930	z2	-930	z”
931	z3	-931	z#
932	z4	-932	z$
933	z5	-933	z%
934	z6	-934	z&
935	z7	-935	z’
936	z8	-936	z(
937	z9	-937	z)
938	z:	-938	z*
939	z;	-939	z+
940	z<	-940	z,
941	z=	-941	z-
942	z>	-942	z.
943	z?	-943	z/
944	{0	-944	{$^{S}_{P}$
945	{1	-945	{!
946	{2	-946	{”
947	{3	-947	{#
948	{4	-948	{$
949	{5	-949	{%
950	{6	-950	{&
951	{7	-951	{’
952	{8	-952	{(
953	{9	-953	{)
954	{:	-954	{*
955	{;	-955	{+
956	{<	-956	{,
957	{=	-957	{-
958	{>	-958	{.
959	{?	-959	{/
960	|0	-960	|$^{S}_{P}$
961	|1	-961	|!
962	|2	-962	|”
963	|3	-963	|#
964	|4	-964	|$
965	|5	-965	|%
966	|6	-966	|&
967	|7	-967	|’
968	|8	-968	|(
969	|9	-969	|)
970	|:	-970	|*
971	|;	-971	|+
972	|<	-972	|,
973	|=	-973	|-
974	|>	-974	|.
975	|?	-975	|/
976	}0	-976	}$^{S}_{P}$
977	}1	-977	}!
978	}2	-978	}”
979	}3	-979	}#
980	}4	-980	}$
981	}5	-981	}%
982	}6	-982	}&
983	}7	-983	}’
984	}8	-984	}(
985	}9	-985	})
986	}:	-986	}*
987	};	-987	}+
988	}<	-988	},
989	}=	-989	}-
990	}>	-990	}.
991	}?	-991	}/
992	~0	-992	~$^{S}_{P}$
993	~1	-993	~!
994	~2	-994	~”
995	~3	-995	~#
996	~4	-996	~$
997	~5	-997	~%
998	~6	-998	~&
999	~7	-999	~’
1000	~8	-1000	~(
1001	~9	-1001	~)
1002	~:	-1002	~*
1003	~;	-1003	~+
1004	~<	-1004	~,
1005	~=	-1005	~-
1006	~>	-1006	~.
1007	~?	-1008	~/
1008	$^{D}_{T}$0	-1008	$^{D}_{T}$$^{S}_{P}$
1009	$^{D}_{T}$1	-1009	$^{D}_{T}$!
1010	$^{D}_{T}$2	-1010	$^{D}_{T}$”
1011	$^{D}_{T}$3	-1011	$^{D}_{T}$#
1012	$^{D}_{T}$4	-1012	$^{D}_{T}$$
1013	$^{D}_{T}$5	-1013	$^{D}_{T}$%
1014	$^{D}_{T}$6	-1014	$^{D}_{T}$&
1015	$^{D}_{T}$7	-1015	$^{D}_{T}$’
1016	$^{D}_{T}$8	-1016	$^{D}_{T}$(
1017	$^{D}_{T}$9	-1017	$^{D}_{T}$)
1018	$^{D}_{T}$:	-1018	$^{D}_{T}$*
1019	$^{D}_{T}$;	-1019	$^{D}_{T}$+
1020	$^{D}_{T}$<	-1020	$^{D}_{T}$,
1021	$^{D}_{T}$=	-1021	$^{D}_{T}$-
1022	$^{D}_{T}$>	-1022	$^{D}_{T}$.
1023	$^{D}_{T}$?	-1023	$^{D}_{T}$/
1024	A@0	-1024	A@$^{S}_{P}$
1025	A@1	-1025	A@!
1026	A@2	-1026	A@”
1027	A@3	-1027	A@#
1028	A@4	-1028	A@$
1029	A@5	-1029	A@%
1030	A@6	-1030	A@&
1031	A@7	-1031	A@’
1032	A@8	-1032	A@(
1033	A@9	-1033	A@)
1034	A@:	-1034	A@*
1035	A@;	-1035	A@+
1036	A@<	-1036	A@,
1037	A@=	-1037	A@-
1038	A@>	-1038	A@.
1039	A@?	-1039	A@/
1040	AA0	-1040	AA$^{S}_{P}$
1041	AA1	-1041	AA!
1042	AA2	-1042	AA”
1043	AA3	-1043	AA#
1044	AA4	-1044	AA$
1045	AA5	-1045	AA%
1046	AA6	-1046	AA&
1047	AA7	-1047	AA’
1048	AA8	-1048	AA(
1049	AA9	-1049	AA)

[a] Positive integer

[b] Negative integer

For integers ± 960 through ± 975, the first character of the parameter is a vertical bar (ADE 124.).

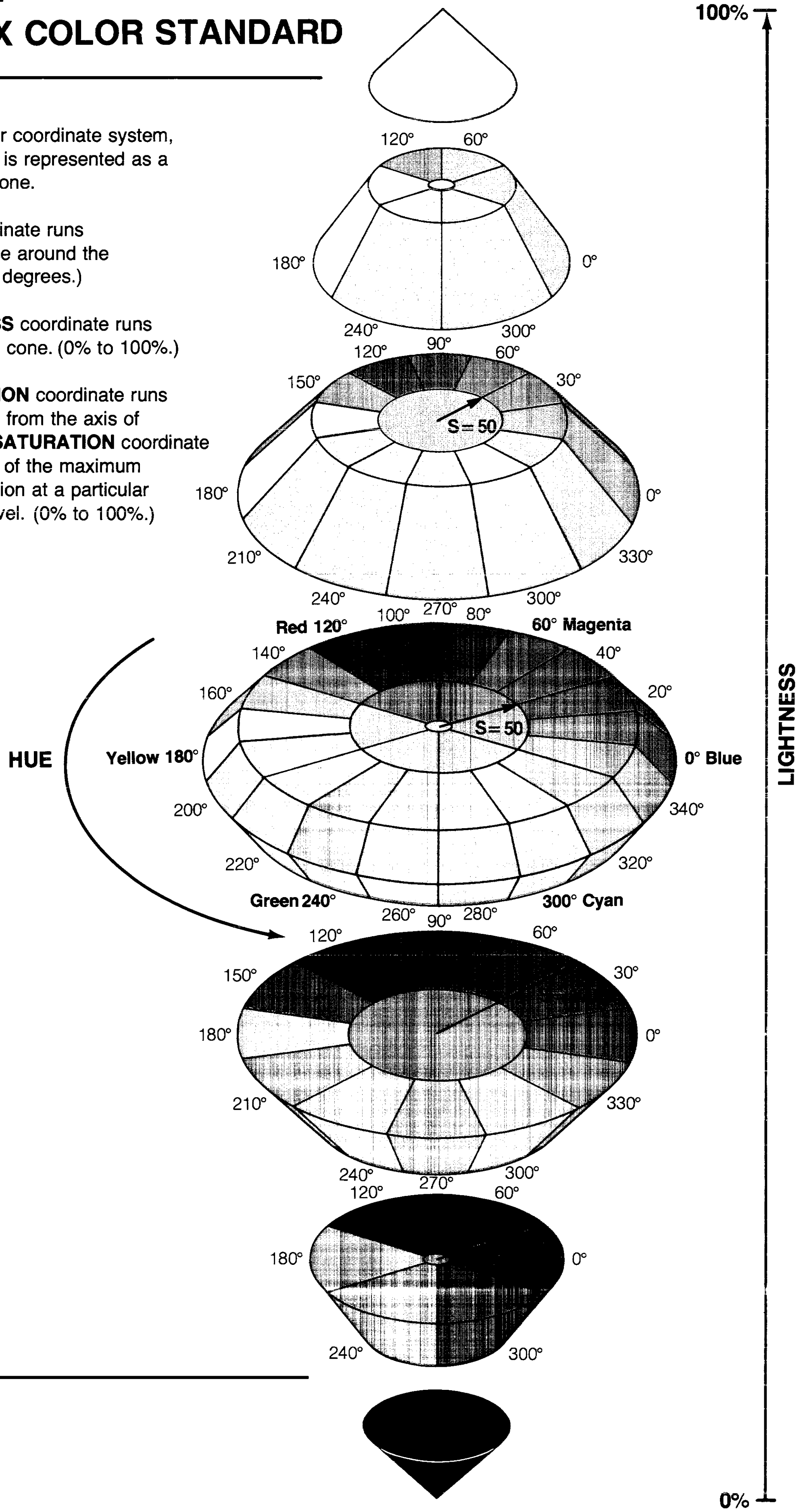
Appendix E
TEKTRONIX COLOR STANDARD

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%.)



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.

INDEX

Tektronix®
COMMITTED TO EXCELLENCE

MANUAL CHANGE INFORMATION

PRODUCT	4107/4109 PROGRAMMERS REFERENCE	CHANGE REFERENCE	C1/1183
MANUAL PART NO.	070-4893-00	DATE	11-23-83

EFFECTIVE ALL SERIAL NUMBERS

TEXT CHANGES

Please take the time to look through this package and make the indicated changes in your manual -- these changes correct errors that may cause you difficulty. Changes to a sentence or paragraph are highlighted by underlining.

THIS IS A PAGE REPLACEMENT PACKAGE.

1. Remove the appropriate pages from your manual and insert the attached pages.
2. Keep this cover sheet in the Change Information section at the very back of this manual for a permanent record.

p.3-12
Insert page 3-12a (at the back of this package) between pages 3-12 and 3-13. Page 3-12a contains a corrected version of Table 3-2; cross out the original version on page 3-12.

p.4-21
In the left column, change the last line to read:

> ... bring the leftmost column into view.

p.5-16
In the left column, just before the fourth paragraph, insert this sentence:

> Another way to define a key in Setup mode is by using the LEARN or NVLEARN command. See the LEARN/NVLEARN command description for details.

p.5-20
Insert page 5-20a (at the back of this package) between pages 5-20 and 5-21. Page 5-20a contains the description of a new command, DIM ENABLE.

p.5-21
In the left column, change the second sentence of the paragraph that begins "DRAW has the same effect ..." to include Edit and VT52 modes. It should read:

> The terminal can execute DRAW when it is in any mode except Ansi, Edit, VT52, or Bypass mode.

In the right column, change the second sentence of the paragraph that begins "DRAW MARKER has the same effect ..." to include Edit and VT52 modes. It should read:

> The terminal can be in any mode except Ansi, Edit, VT52, or Bypass mode when it executes DRAW MARKER.

In the right column, replace the last paragraph with this one:

> The terminal always draws markers in Overstrike mode. See the SET GRAPHICS AREA WRITING MODE command for details about how the Overstrike writing mode operates.

p.5-22

In the left column, first sentence of the last paragraph, add Edit, and VT52 after the word "Ansi" so that the sentence begins "In Ansi, Edit, and VT52 modes, the terminal ..."

p.5-31
In the right column, fourth paragraph, put a period after the word "results", and delete the rest of the paragraph.

p.5-33
In the right column, first paragraph following the syntax box, change the second sentence to read:

> A key programmed with **NVLEARN** and saved with **NVSAVE** (SAVE NONVOLATILE PARAMETERS)
> remains programmed ...

p.5-35
In the left column, delete references to a keyword in Setup syntax from the parameter description so that it reads:

```
locking-mode: integer; one of the following:
   0   Unlocks the viewing keys
   1   Locks the viewing keys
Defaults: Factory  = 0
          Power-Up = 0
          Omitted  = 0
```

p.5-37
In the left column, delete the last item in the bulleted list (the item that begins "Cancels a command ...").

p.5-44
In the left column, second paragraph, change the word integer to integer-report.

In the left column, change Bit 3 (B3) of Table 5-9 from **X** to **C** and change the list of bit meanings to read:

X	Reserved for the future
C	The copier is connected and powered up
B	The copier is busy
P	The copier interface is present

p.5-50
In the left column, delete the words "or screen" from the sentence that precedes the syntax box.

p.5-56
In the left column, delete both occurences of "EOM-indicator" in the indented lists.

p.5-57
In the left column, delete the last paragraph.

p.5-58
In the right column, expand the next-to-last paragraph to read:

> This command saves only parameters that have changed since the last time this command was issued. Additionally, this command saves any macros defined with a DEFINE NONVOLATILE MACRO or NVLEARN command.

P.5-62
In the left column, delete the words small integer from the paragraph that begins with "font-code:".

Following that paragraph, add this new paragraph:

> The character sets that are the current G0 and G1 sets can be selected with the Ansi mode command SCS (Select Character Set).

p.5-63
In the left column, change the Setup name in the Setup Syntax box from **BACKINDICES** to **BACKINDEX**.

p.5-64
Insert page 5-64a (at the back of this package) between pages 5-64 and 5-65. Page 5-64a contains a new version of the parameter value list for the SET BORDER VISIBILITY command.

p.5-70
In the left column, in the paragraph that begins "color-mixture:", delete the sentence that begins "Unlike other integer arrays ..."

In the right column, delete the integer **4** from the Setup syntax example so the example reads:

DACMAP 3 0 100 0

Immediately following this example, delete the sentence that begins "The **4** indicates ..."

p.5-74
Insert page 5-74a between pages 5-74 and 5-75. Page 5-74a contains a new version of the SET EDIT CHARACTERS command. Cross out the old version on page 5-74.

p.5-75
In the left column, second paragraph after the Syntax boxes, delete the sentence that begins "When the terminal reaches the end ..."

p.5-76
In the left column, change the sentence that precedes the Host Syntax box to read:

> Specifies the character(s) used to terminate lines of text that the operator types.

In the left column, second-to-last paragraph, change the first sentence to read:

> For example, in Prompt mode, when the operator types one of the end-of-message character(s), the terminal stops transmitting ...

p.5-84
In the left column, change the Setup name in the Setup Syntax box from **GINRUBBERBANDING** to **GINRUBBERBAND**.

p.5-95
In the left column, change the Setup name in the Setup Syntax box from **GINAPERTURE** to **GINPICKAPERTURE**.

p.5-98
In the right column, add 19,200 to the list of valid baud rates found in the baud-rate parameter description.

p.5-103
In the right column, delete the first two paragraphs.

p.5-116
Insert page 5-116a between pages 5-116 and 5-117. Page 5-116a contains the description of the new command SET TABLET HEADER CHARACTERS.

p.5-118
In the left column, change the Setup name in the Setup Syntax box from **VIEWATTRIBUTES** to **VATTRIBUTES**.

p.5-119
In the left column, change the Setup name in the Setup Syntax box from **VDISPLAYCLUSTER** to **VCLUSTER**.

p.B-2
Add this error code:

IH11 (Level 2). Invalid parameter (must be 0 or 1; in Setup syntax must be **letters** or **control**).

p.B-3
Add these error codes:

JQ11 (Level 2). Invalid device specifier.

JQ12 (Level 3). Out of memory while parsing the first parameter.

KG11 (Level 2). Invalid dim-code (must be 0 or 1).

KH00 (Level 2). Unrecognized command (you cannot make a small dialog copy with Katakana Option 4K).

p.B-4
Change these error codes to read as follows:

MC11 (Level 2). Invalid value in parameter 1 (must be in range 0 through 4095).

MC21 (Level 2). Invalid value in parameter 2 (must be in range 0 through 4095).

MC31 (Level 2). Invalid value in parameter 3 (must be in range 0 through 4095).

p.B-5
Change these error codes to read as follows:

NU11 (Level 2). Invalid numeric equivalent of bypass-cancel character (must be in range 0 through 127).

PA11 (Level 2). Invalid port-identifier (must be **P0:** or **P1:**), or port is busy.

PB11 (Level 2). Invalid port-name (must be **P0:** or **P1:**), or port is busy.

p.B-6
Change these error codes to read as follows:

> **PE11 (Level 2).** Invalid port-name (must be PO: or P1:), <u>or port is busy</u>.
>
> **PI11 (Level 2).** Invalid port-name (must be PO: or P1:), <u>or port is busy</u>.
>
> **PR11 (Level 2).** Invalid port-name (must be PO: or P1:), <u>or port is busy</u>.
>
> **PR21 (Level 2).** Invalid baud rate (must be 75, 110, 134, 150, 300, 600, 1200, 1800, 2000, 2400, 4800, 9600, or <u>19200</u>).

Delete error code **PL23.**

p.B-9
Add this error code:

> **SK13 (Level 2).** Segment specified is an active GIN cursor.