Data General

ABOUT THE COVER This unretouched photo vividly demonstrates the power of the DESKTOP GENERATION's optional color display. A Data General MV/8000 computer was used to convert the digitized image into the 16-color format used by the DESKTOP GENERATION.

Data General

Programmer's Referenc

DESKTOP GENERATION™

Model 10 and 10/SP System Console

Notice

Ordering No. 014-000770

Preface

This manual describes the system console keyboard and display of the DESKTOP GENERATION™ Model 10 and 10/SP computer systems. Other technical and programmer's references for the DESKTOP GENERATON are listed and summarily described under "Related Manuals" in this preface.

Organization

- Chapter 1 gives an overview of the console's features.
- Chapter 2 gives a detailed description of the console keyboard.
- Chapter 3 describes the monochrome display, with a detailed reference on its command functions.
- Chapter 4 describes the color display, with a detailed reference on its command functions.
- Appendix A lists the character codes used by the keyboard and display.
- Appendix B gives several programming examples in higher-level languages.
- Appendix C describes how to program the display in assembly language. (This information is of interest mainly to advanced programmers.)
- Appendix D lists the physical and electrical specifications of the console.
- The index alphabetically lists the concepts and terms in this book and references the pages on which they appear.
- Last, a publications comment form invites you to help Data General improve future publications by evaluating this manual.

Related Manuals

A comprehensive documentation set supports all the hardware and software products available for DESKTOP GENERATION computers. The hardware-related books listed below fall into three categories: the technical reference series; the user guides for operating, installing, and testing; and the introductory guide for DESKTOP GENERATION computers.

The following technical and programmer's references address the needs of assembly language programmers and engineers.

16-bit Real Time ECLIPSE Assembly Language Programming

Global in nature, this book explains the processor-independent concepts, functions, and instruction sets of 16-bit ECLIPSE computers. DGC ordering no. 014-000688.

Model 10 and 10/SP Computer Systems
Technical Reference

In addition to the functional and physical organization of Model 10 and 10/SP computers and their technical specifications, this book explains their processor-unique concepts, functions, and instruction set features. Provides detailed information for programming the systems' I/O devices, including the diskette subsystem, and explains the theory of operation for the basic components of Models 10 and 10/SP. DGC ordering no. 014-000766.

Model 20 and 30 Computer Systems
Technical Reference

In addition to the functional and physical organization of Model 20 and 30 computers and their technical specifications, this manual explains their processor-unique concepts, functions, and instruction set features. Also included are guidelines for programming the I/O devices, including the diskette subsystem, and a theory of operation for the basic components of Models 20 and 30. DGC ordering no. 014-000767.

I/O and Interfacing
Technical Reference

Introduces the microI/O bus and describes the I/O interface required to communicate with this bus and its host DESKTOP GENERATION computer. Discusses the I/O instruction set and the I/O program interrupt and data channel facilities. Includes a chapter about the 4210 general-purpose interface, useful to those designing a custom I/O interface for their system. DGC ordering no. 014-000774.

For more detailed information about the microI/O bus and Data General integrated circuits used in the I/O interface, refer to *microNOVA Integrated Circuits Data Manual.* DGC ordering no. 014-000074.

Communications Interfaces
Technical Reference

Discusses the functional and physical organization of the asynchronous/ synchronous communications interfaces available for DESKTOP GENERATION computers. Defines their I/O instruction sets, offers guidelines for writing assembly language I/O subroutines, and contains theory of operation for each communications card. DGC ordering no. 014-000769.

Sensor I/O
Technical Reference

Defines instruction sets, offers guidelines for writing assembly language I/O subroutines, describes theory of operation at an overview level, and explains how to connect field wiring for the 4222 digital I/O interface, 4223 analog-to-digital interface, 4224 digital-to-analog interface, and 4335 analog subsystem. DGC ordering no. 014-000775.

Model 6271 Disk Subsystem
Technical Reference

Describes the Model 6271 disk subsystem that supplies up to 30 megabytes of on-line storage for DESKTOP GENERATION computers. DGC ordering no. 014-000768.

IEEE-488 Bus Interface
Technical Reference

Provides the information needed to interface, program in assembly language, and troubleshoot this card in a DESKTOP GENERATION system. Reviews the contents of the IEEE-488 bus standard, summarizing its commands, messages, and states, and includes a theory of operation. DGC ordering no. 014-000773.

The following books are how-to manuals written for anyone who needs to know how to install, operate, and test a DESKTOP GENERATION system.

Installing Model 10 and 10/SP Systems

The first book that a Model 10 or 10/SP owner should read, explains how to unpack and install either system and its optional peripherals. Simple instructions and ample illustrations make the book accessible to any reader. DGC ordering no. 014-000901.

Operating Model 10 and 10/SP Systems

A logical follow-on to Model 10 and 10/SP installation, this guide takes you from powering up the system and its optional peripherals through performing such routine operations as loading paper in a printer and inserting or removing diskettes. Brings you to the point of loading the system software. Amply illustrated and written for users at any level of experience. DGC ordering no. 014-000900.

Testing Model 10 and 10/SP Systems

Follows the installation and operating manuals with instructions for verifying the operation of Model 10 or 10/SP systems and their optional peripherals. Steps you through the power-up test and Customer Diagnostics and explains how to troubleshoot customer-replaceable components. Simple instructions and diagrams make the book accessible to any user. Includes phone numbers for Data General assistance. DGC No. 014-000902.

Installing Model 20 and 30 Systems

The first book a Model 20 or 30 owner should read, explains how to unpack and install either system and its optional peripherals. Accessibly written and illustrated, for users at any level of experience. DGC ordering no. 014-000904.

Operating Model 20 and 30 Systems

Follows Model 20 and 30 installation, leading you from powering up the system and its optional peripherals through performing such routine operations as loading paper in a printer and inserting or removing diskettes. Brings you to the point of loading the system software. The simple instructions and generous illustrations are suitable for any reader. DGC ordering no. 014-000903.

Testing Model 20 and 30 Systems

A follow-on to the installation and operating mauals, explains how to verify the operation of Model 20 or 30 systems and their optional peripherals. Simple instrctions and diagrams lead you through the power-up test, Customer Diagnostics, and trouble-shooting of customer-replaceable components. Includes phone numbers for Data General assistance. DGC ordering no. 014-000905.

This last book is a product overview, addressed to all DESKTOP GENERATION users.

The DESKTOP GENERATION

Introduces the DESKTOP GENERATION, summarizing each model of the family, and describes its many hardware and software products, features, and capabilities. Includes a brief history of Data General, a sampling of applications, and an overview of the customer service and support programs available to you as a Desktop Generation user. DGC ordering no. 014-000751.

Contents

3 Monochrome Display

Introduction 1

This chapter introduces the system console. It gives an overview of the keyboard and display features, and some basic instructions on operating the unit.

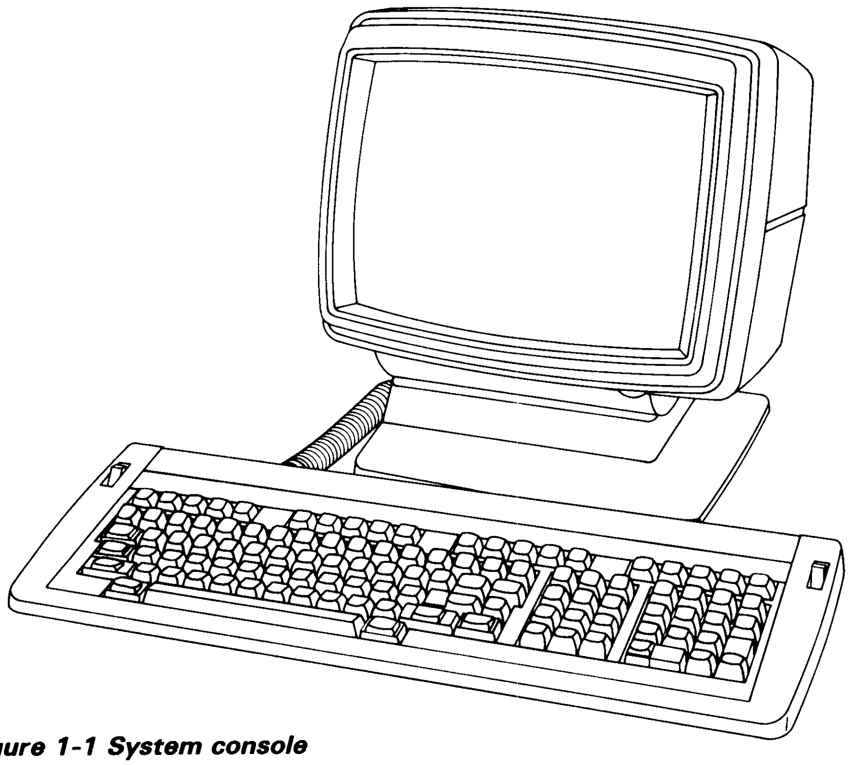


Figure 1-1 System console

Overview

The system console, with its keyboard and display, is perhaps the most important peripheral of any desktop computer. People use it to communicate directly with the computer, in words and pictures, often for many hours at a time. The console must be powerful, easy to understand, and friendly in everyday use.

The DESKTOP GENERATION Model 10 and 10/SP system console meets all of these goals. It is an integral part of the computer, not a separate terminal, so it provides many advanced features at a low cost.

Keyboard

The console keyboard contains 107 keys, and is connected to the computer by a cable, so you can easily move it to a comfortable position on your desk or your lap. Its human-engineered design has tilt adjustments and high-quality key switches for reliable, effortless typing.

The keyboard's features include:

- A main typewriter-style keyboard with 61 keys, including computer functions such as NEW LINE and BREAK.
- A 14-key numeric keypad with decimal point, comma, minus sign, and NEW LINE (Enter) keys.
- A 12-key screen management keypad for cursor movement, printing, and erasing, plus four user-definable keys.
- Fifteen user-definable function keys, arranged in three groups of five, with a slot to hold identifying labels.

The keyboard also contains four indicator lights and a signal beeper. It is available in eight different versions: American, United Kingdom, and seven European styles.

Display

The console display is available in either a monochrome or color version. Both can display text in 24 rows of 80 characters, and graphic images with a resolution of 640 by 240 points. Text and graphics can be freely intermixed on the screen.

When the screen is in graphic mode, it is controlled by a built-in interpreter program that makes it easy to control the display with any programming language. You can send simple words such as LINE, CLEAR, and COLOR to the screen, and the interpreter performs the specified functions. More advanced programmers can use assembly language routines to control the screen, for high speed animation and other special effects.

With its optional color display, the DESKTOP GENERATION Model 10 and 10/SP become the most powerful graphics system available in a desktop unit. It can create pictures using a *palette* of sixteen colors. Each of these sixteen colors can be chosen from a total spectrum of 4096 different hues and shades.

The color option also allows characters to be displayed in sixteen colors. Text and pictures are kept in separate memories, so either one can be written and erased without destroying the other. The text colors, and other character features such as underline and blink, are compatible with the IBM PC.

Operation

As mentioned above, the console is an integral part of the computer. However, the system's hardware and built-in software enable you to program the console as if it were a separate terminal. With each computer, Data General supplies a D/200 emulator program that enables the console to support all the features of the popular D/200 terminal. This ensures that the Model 10 or 10/SP console is compatible with software written for other systems.

Power-up

When you turn on the power to your computer, it automatically runs its self-test program. The indicator lights on the keyboard blink, and the signal tone sounds several times, while the computer types a short message on the display. Upon successful completion of the self-test, the computer displays the exclamation point prompt, indicating that it is ready to accept a command from the keyboard.

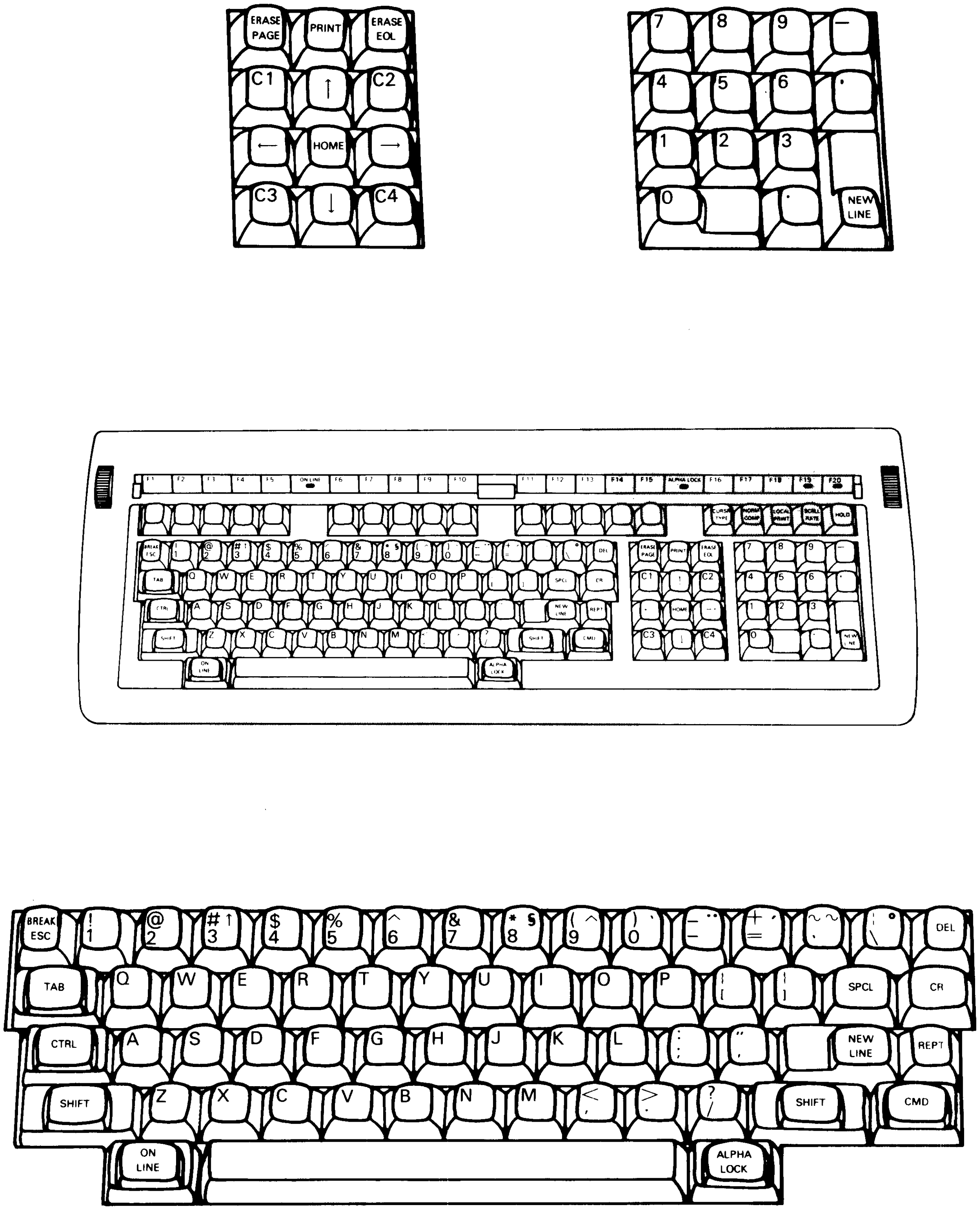
If the exclamation point does not appear, your computer has failed its self-test. In that case, you may consult your testing manual for instructions on how to identify and solve the problem.

When the computer displays the prompt, the console is in a mode which provides a minimum set of display features: upper case letters only, and basic control character functions such as NEW LINE, ERASE PAGE, and cursor movement. At this time you will probably want to load the D/200 emulator, to provide your applications with a more powerful display.

The procedure for loading the D/200 emulator depends on what type of software you are using with the computer. In some cases, the emulator is supplied on a self-booting diskette. In other cases, loading the emulator is one step of a larger installation procedure. Consult your software installation guide for specific instructions.

Keyboard 2

This chapter describes the system console keyboard. It contains details on use of the different keypads and the special purpose keys. For a list of the codes generated by the keys, see Appendix A.



DG-25239

Figure 2-1 System console keyboard

Organization

The Model 10 and 10/SP keyboard contains 107 keys in a versatile, human-engineered package (see Figure 2-1). Functionally, the keys may be divided into two categories. The *code-generating* keys generate one or more ASCII codes when pressed. The *modifying* keys do not generate any code by themselves: they modify the functions of other keys.

Most of the code-generating keys have a *typematic* repeat feature. If a key is held down for more than 3/4 second, it begins automatically repeating at a rate of six times per second. (For faster repeating action, you can use the REPT key described below.)

The ASCII codes that are generated by the various keys are summarized in Appendix A.

NOTE *Throughout this chapter, octal numbers enclosed in angle brackets represent ASCII character codes. For example, <040> is the code for a space (40 octal = 32 decimal).*

Modifying Keys

The most commonly used modifying key is SHIFT, with which you are probably familiar. You use it by holding it down while typing another key. SHIFT forces all letter keys to generate upper case instead of lower, and it selects the upper symbol for any keycap that shows two symbols. For keycaps with three symbols, SHIFT selects the upper-left symbol. (The upper-right symbol is accessed with the SPCL key, described below.)

The CTRL (pronounced "control") key is used like SHIFT, by holding it down while pressing another key. It causes all letter keys, and some others, to generate the corresponding control characters. Some of these have standard functions that are equivalent to other keys: for instance, Control-J is equivalent to NEW LINE, and Control-L is equivalent to ERASE PAGE. You can use CTRL and SHIFT together; the user-defined function keys, for instance, can each generate one of four different codes, depending on whether they are pressed in conjunction with SHIFT, CTRL, or both.

The CMD (pronounced "command") key only works with a few keys. It is a sort of "safety SHIFT" for functions that you will want to avoid typing accidentally. For example, holding CMD while typing BREAK/ESC interrupts the computer.

The SPCL (pronounced "special") key allows you to generate foreign language characters and other special symbols. To use SPCL, you first press it *by itself*, and then press the key to be modified. This is how you access the upper-right symbol on keys with three symbols. For example, type SPCL, then type 8. Instead of the digit 8, your screen will display the paragraph symbol that is shown on the upper-right corner of the keycap.

In some cases, you use SPCL with *two* additional keys to generate a code. For example, just above and to the left of SPCL is a key showing the *tilde* symbol. Press SPCL followed by the tilde key: you will note that no code is generated. Now press the space bar, and the tilde symbol will appear. Next, press SPCL and then tilde, followed by a letter N (either upper or lower case). Your screen will show a Spanish-style N with the tilde directly over it. The keys following SPCL may be in either order; SPCL-tilde-N and SPCL-N-tilde are equivalent.

The ALPHA LOCK key locks all letter keys so that they always generate upper case, even when you do not use SHIFT. Pressing this key once locks the letter keys; pressing it again unlocks them. The indicator light, located above the PRINT key, turns on whenever ALPHA LOCK is in effect.

The REPT (repeat) key provides a high-speed repeat function for any character. Holding this key down will cause any key that you type to repeat at a rate of 15 characters per second. You can use REPT in conjunction with CTRL and SHIFT, to generate a stream of shifted or control characters.

Main Keypad

The main keypad contains 61 keys that are organized in a standard typewriter-style layout, with a number of special purpose keys at the left and right sides.

The special purpose keys in the main keypad include:

BREAK/ESC Pressing this key by itself generates the ASCII "escape" character, <033>. Pressing this key while holding CMD interrupts the computer, and causes it to run its virtual console program. If you interrupt a program by mistake, type P to resume normal operation.

TAB Pressing this key generates the code <011>. This code is used as a horizontal tab function on most systems.

DEL Pressing this key generates the DELETE or RUB OUT code, <177>. On most systems, this code is used to delete the last character typed.

ON LINE This key has no function, since the Model 10 or 10/SP console is always on line when the power is turned on.

Screen Management Keypad

This keypad contain keys for cursor control and screen erasing, as well as five user-definable function keys. The key functions are:

Arrow Keys These generate the codes for the Cursor Left <031>, Right <030>, Up <027>, and Down <032> display commands. They move the cursor one character position in the specified direction. When you press one of these keys while holding SHIFT, it generates the function header <036>, followed by the regular code.

HOME This key generates the code for the Home Cursor <010> display command, which moves the cursor to the "home" position at the upper left corner of the screen.

ERASE PAGE This key generates the code for the Erase Page <014> display command, which erases the entire screen, and returns the cursor to the home position. Pressing this key while holding CMD resets the display to its power-up condition; this is equivalent to the Reset display command. (It does not reset the entire machine, only the display.)

ERASE EOL This key generates the code for the Erase to End of Line <013> display command, which erases the part of the screen from the current position of the cursor to the right margin.

PRINT When pressed by itself, this key generates no code. When you press it while holding CMD, this key generates two codes: the function header <036> followed by <021>. When pressed while you hold CMD and SHIFT, this key generates <036> <001>.

C1, C2, C3, and C4 These are user-definable keys, whose function may be controlled by your application. Each of these keys generates two codes when pressed: the function prefix <036> followed by an identifying code. Holding down SHIFT while pressing one of these keys generates a different identifying code, so a total of eight functions are available.

Numeric Keypad

The keys in the numeric pad have the same function as the corresponding keys in the main keypad. The calculator-style layout, with one-handed access to decimal point, minus, and NEW LINE keys, makes it easy to enter large amounts of numeric data with this keypad.

User-Defined Function Keys

There are fifteen function keys, labelled F1 through F15, arranged along the top of the keyboard in three groups of five. (The unlabelled keys in the fourth group are reserved for future use by Data General, and do not generate any codes.)

Each of the keys F1 through F15 generates the function header <036> followed by an identifying code. Each key can generate one of four different identifying codes, depending on whether you type it by itself, with SHIFT, with CTRL, or with both SHIFT and CTRL. Thus a total of 60 different functions are available for your application. A slot in the keyboard housing allows you to place identifying labels over the keys.

Other Keyboard Features

The keyboard housing contains four indicator lights. One, located above the 5 key, shows that the power is on. Another, located above the PRINT key, indicates when the keyboard has been placed in upper case mode by the ALPHA LOCK key. The other two are reserved for future use by Data General.

The keyboard also contains a signal beeper. It produces a short tone whenever you send a Bell <007> character to the display.

Monochrome Display 3

This chapter describes the monochrome display. It contains a general description of the display features, and a detailed reference on the various commands.

The monochrome display can produce both characters and high resolution graphics. It permits you to mix text and pictures on the screen, as shown in Figure 3-1.

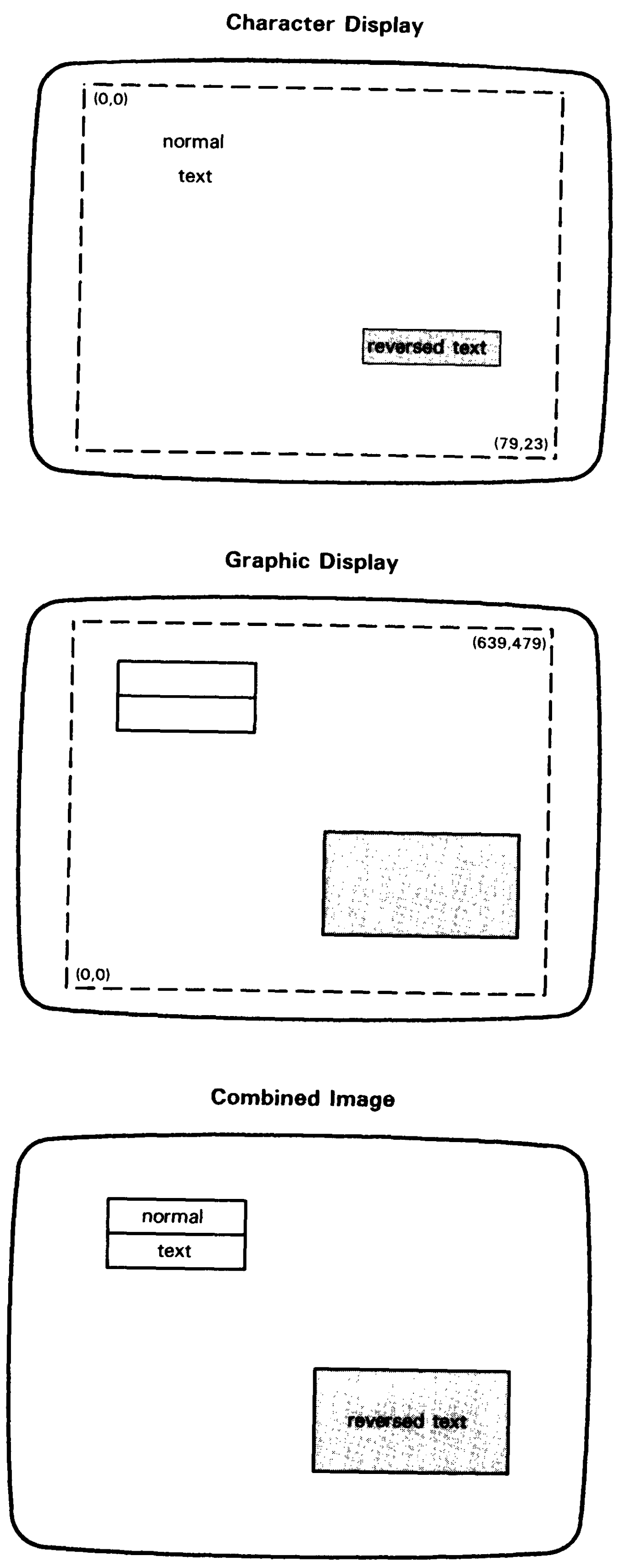


Figure 3-1 Monochrome-screen

ID-00833

Character Display

The screen can display 24 lines of 80 characters. Character positions are referenced by their X and Y coordinates, which are the column and line number. The upper left corner has coordinates (0, 0); the lower right has coordinates (79, 23).

The computer can generate 256 different characters, including the standard ASCII set, and a number of foreign language symbols. You can use the Define Character command to create your own characters, either by redefining the existing ones or by giving definitions to the unused ones.

Each character position has several *attributes* that you can control. They are:

Foreground and background colors: black or green.

Video mode: normal or reversed.

Brightness: normal or dim.

Blinking: on or off.

Underline: on or off.

The computer maintains a *cursor* to mark the position at which the next character will be written. You can select several different cursor shapes, or make it invisible, with the Set Cursor Type command. By sending various control characters to the display, you can move the cursor to any position on the screen. A program can also interrogate the screen, to read the cursor's current position.

Graphic Display

The graphic display consists of a matrix of points called *pixels*. The screen is 640 pixels wide and 240 pixels high. The computer draws pictures by setting individual pixels to black or green. Since even a simple drawing can involve hundreds or thousands of pixels, the console has built-in commands that help you to draw lines, boxes, and text. *Style* commands permit you to easily create dotted or dashed lines, and boxes with stripes or other texture patterns.

To design a graphic image, you must specify the X and Y coordinates on the screen where various objects are to be drawn. The lower left corner of the screen has coordinates (0, 0); the upper right corner has coordinates (639, 479). Note that, although the screen is physically 240 pixels high, the console firmware defines it to be 480 pixels high. If you try to write to a pixel with an odd-numbered Y coordinate, the console will actually write to the next lower even-numbered pixel. For example, an attempt to plot a point at location (345, 123) will be directed to (345, 122); the Y coordinate is converted, but the X coordinate is unchanged.

The graphic display has a cursor that is used to write or read any pixel. Unlike the character cursor, the graphic cursor is not visible. However, the computer keeps track of its location for use by various graphic commands.

Color Selection

The color of a pixel is controlled by its value, which is a single bit. Pixels containing 0 are displayed in black, and pixels containing 1 are displayed in green.

One of the two colors is always specified to be the *background*, and the other is the *foreground*. At power-up, color 0 (black) is selected as the background. You can select a green background with the Clear command. The choice of colors affects the operation of the graphic and character output commands, since they can draw in both foreground and background colors.

Although only two colors are available, the display actually accepts values from 0 to 15 in commands such as Set Drawing Color. This ensures that programs written for the color display (see Chapter 4) will run on a monochrome system without producing errors. The console converts all non-zero color values to 1 (green).

Drawing Functions

Lines and Points

The console provides two commands for line drawing. To use the Draw Line command, you specify the start and end points of the line. To use the Attach Line command, you specify only the end point; the starting point is the current position of the graphic cursor. The Attach Line command is convenient when you are drawing a complex shape with a series of connected line segments.

To draw dotted or dashed lines, you use the Set Line Style command to specify a pattern of *on* and *off* pixels. When you draw a line, the *on* and *off* pixels are set to different colors. The specific colors that are used depend on the current drawing color, background color, and on the drawing mode as described below.

The console provides a Draw Point command to change the color of a single pixel. This is equivalent to drawing a line with its start and end point at the same location.

Boxes

The console provides a Fill Box command for drawing rectangular boxes of any size. Boxes may be drawn in a solid color, or in a texture pattern such as stripes or checks.

To draw textured boxes, you use the Set Box Style command to specify a pattern of *on* and *off* pixels. When you draw a box, the *on* and *off* pixels are set to different colors. The specific colors that are used depend on the current drawing color, background color, and on the drawing mode as described below.

Text

For maximum freedom in combining text and pictures, the console has a Draw Text command that lets you write characters in graphic mode. Text that is written in this manner can be positioned anywhere on the screen; it need not fit exactly into the 80-by-24 grid.

To draw a character, the console reads the pattern of *on* and *off* pixels that is stored in the character set memory. This is the same memory that is used for the character display. You can define new character shapes, or redefine old ones, with the Define Character command.

When you draw a character, the *on* and *off* pixels are set to different colors. The specific colors that are used depend on the current drawing color, background color, and on the drawing mode as described below.

Drawing Modes

The console's drawing mode commands provide three different ways of drawing the *on* and *off* pixels that are specified by the line style, box style, or character shape. The three modes are called Set, Replace, and XOR (pronounced "ex-or"). In Set mode, which is selected at power-up, the *off* pixels are left in their previous color. For example, if you draw a dashed line across a picture, the spaces between the dashes will be unchanged.

In Replace mode, the *off* pixels are set to the current background color. For example, if you draw a dashed line across a picture, the spaces between the dashes will be set to the background color.

In XOR mode, the *off* pixels are left unchanged. The *on* pixels, however, are set to the complement of their previous value: that is, 0 pixels are changed to 1, and vice versa.

XOR mode produces some odd effects, but it has the unique advantage that you can "undraw" any part of a picture by simply drawing it twice. This effect is most useful for programs that display temporary items, such as a menu or cursor, in front of a picture. The temporary material can be quickly removed, and your program will not need to laboriously redraw the original picture.

Programming

The simplest way to control the display is by simply sending characters to it. In addition to the "printing" characters (letters, digits, and other symbols), there are a number of characters that have special control functions. Some of these control characters, such as NEW LINE and ERASE PAGE, are common to most terminals that use the ASCII character set. Other characters have functions such as moving the cursor, changing colors or attributes, and defining new characters. Of particular importance is the Enter Graphic Mode character sequence, since it changes the way the display handles other characters, and provides easy access to the graphic functions.

When using the console, you should remember that the character and graphic data are in the same memory, so there may be some interaction between them. In particular, the character attributes, such as blink and dim, will affect graphic images.

In general, character commands do not work in graphic mode, and vice versa. If your program uses both, you must remember to enter and leave graphic mode, depending on which type of command you want to use next.

Interpreter Mode

The graphic interpreter is a program that is provided with every DESKTOP GENERATION Model 10 and 10/SP computer. Its function is to read text commands such as POINT, LINE, etc., and perform the specified functions. This makes it very easy for you to create pictures in any programming language, including command languages such as the Data General CLI.

To activate the interpreter, you send an Enter Graphic Mode character sequence to the display. When the console is in graphic mode, all characters that you send to it are interpreted as graphic commands. The commands consist of keywords and numbers. The numbers that specify the parameters for a command must be sent *before* the keyword. For example, to draw a point at pixel position (123, 456) send the text

```
123 456 POINT
```

to the display. To draw a line from the lower left corner of the screen (coordinates 0, 0) to the upper right corner (coordinates 639, 479), send

```
0 0 639 479 LINE
```

When you finish drawing a picture, you may send a Leave Graphic Mode character sequence to the display. This deactivates the graphic interpreter, so that all characters that you send to the screen will be displayed in the normal manner.

In the above examples, the numbers and words are separated by spaces. In fact you can use one or more of the space, comma, or NEW LINE characters as separators between parts of a command.

When you send commands to the interpreter, it stores them in a memory buffer. A command is not actually executed until either the buffer becomes full, or you send a NEW LINE character. Therefore, to ensure that all commands are executed, you should always send a NEW LINE to the screen before leaving graphics mode.

Although the above examples use only upper case letters for commands, the console accepts either upper or lower case.

Assembler Language Access

For advanced programs, or those that must run at high speed, you can program the display in assembler language. This eliminates the need for the graphic interpreter to read and interpret text commands. Instead, your program can jump directly into the computer's firmware by using the EHYP instruction. Assembler language access is described in detail in Appendix C. More detailed information on the computer may be found in the *Model 10 and 10/SP Computer Systems Technical Reference*, DGC no. 014-000766.

Command Summary

The following conventions are used to describe display command formats:

- Numbers enclosed in angle brackets represent the character codes that you send to the display to perform a function. For example, Erase Page is defined as <014>, so to erase the screen, you must send a control-L (14 octal, 12 decimal) to the display.
- Letters enclosed in angle brackets represent character codes whose values depend on the application. For more information, consult the Reference section later in this chapter.
- Upper case sans serif words are text that should be sent to the display exactly as shown. For example, to select color 1 for drawing in the graphic plane, you would send the text 1 COLOR. (Note that you may actually use either upper or lower case in your programs; this manual uses only upper case for clarity.)
- Lower case letters represent numbers that may have a range of values. Specifically:

  x, x1, and x2 represent horizontal coordinates in the graphic plane, and may range from 0 to 639.

  y, y1, and y2 represent vertical coordinates in the graphic plane, and may range from 0 to 479.

  c represents a pixel color value, and may range from 0 to 15. The console converts all non-zero values to 1.

  f represents a number used as a logical flag. A value of zero means False or Off, and any other value means True or On.

  n, n1, n2, etc., represent arbitrary numbers whose value depends on the specific application. For more information, consult the Reference section later in this chapter.

Graphic Output Commands

Command	Form	Function
Attach Line	x y LINETO	Draw a line from current position to specified point.
Clear	c CLEAR	Clear screen and select background color.
Draw Line	x1 y1 x2 y2 LINE	Draw a line between the specified points.
Draw Point	x y POINT	Draw a single point.
Draw Text	'text' or "text"	Write characters on the graphic plane.
Erase Page	<014>	Erase entire screen and reset cursors and character modes.
Erase to End of Line	<013>	Erase screen from cursor position to end of line.
Erase to End of Page	<036><106> <106>	Erase screen from cursor position to end of page.
Fill Box	x1 y1 x2 y2 FILLBOX	Draw and fill a rectangular box at the specified location.
Reset	<036><106> <101> or COLD	Clear entire screen, reset console to power-on state.
Set Drawing Color	c COLOR	Select color for graphic commands.
Set Fill Style	n1 ... n16 FSTYLE	Specify texture pattern for filling boxes.
Set Graphic Cursor Position	x y CURRENT	Set position of graphic cursor.
Set Line Style	n LSTYLE	Set pattern for line drawing (solid, dotted, dashed, etc.)
Start Replace Mode	REPLACE	Use Replace drawing mode for graphics.
Start Set Mode	SET	Use Set drawing mode for graphics.
Start XOR Mode	XOR	Use XOR drawing mode for graphics.

Character Output Commands

Command	Form	Function
Define Character	<036><106> <122> <n1>... <n10>	Define a new symbol for a character.
End Blink	<017> 15_{10}	Write subsequent characters with blink attribute off.
End Dim	<035>	Write subsequent characters with normal brightness.
End Reverse Video	<036><105> 30_{10} 69_{10}	Write subsequent characters with normal fore/background.
End Underline	<025>	Write subsequent characters without underlines.
Erase Page	<014>	Erase entire screen and reset cursors and character modes.
Erase to End of Line	<013>	Erase screen from cursor position to end of line.
Erase to End of Page	<036><106> <106>	Erase screen from cursor position to end of page.
Reset	<036><106> <101> or COLD	Clear entire screen, reset console to power-on state.
Set Character Background Color	<036><102> <n>	Set background color for subsequent characters.
Set Character Foreground Color	<036><101> <n>	Set foreground color for subsequent characters.
Start Blink	<016> 14_{10}	Write subsequent characters with blink attribute on.
Start Dim	<034> 28_{10}	Write subsequent characters at low intensity.
Start Reverse Video	<036><104> 30_{10} 68_{10}	Write subsequent characters with reversed fore/background colors.
Start Underline	<024>	Write subsequent characters with underlines.

Cursor Control Commands

Command	Form	Function
Carriage Return	<015>	Return cursor to left margin.
Cursor Down	<032>	Move cursor down one position.
Cursor Left	<031>	Move cursor left one position.
Cursor Right	<030>	Move cursor right one position.
Cursor Up	<027>	Move cursor up one position.
Enter Graphics Mode	<036><107>	Activate graphic interpreter.
Erase Page	<014>	Erase entire screen and move cursor to upper left corner.
Home Cursor	<010>	Move cursor to top left corner of screen.
Leave Graphics Mode	<036><120>	Deactivate graphic interpreter.
New Line	<012>	Move cursor to start of next line.
Read Cursor Position	<005>	Request cursor position report from console.
Reset	<036><106> <101> or COLD	Clear entire screen, reset console to power-on state.
Set Cursor Position	<020><n1> <n2>	Move character cursor to specified location.
Set Cursor Type	<036><106> <121><n>	Select shape of cursor.
Set Graphic Cursor Position	x y CURRENT	Move graphic cursor to specified location.

Status and Control Commands

Command	Form	Function
Disable Blink	<004>	Disable blinking for the whole screen.
Disable Roll	<023>	Turn off automatic scrolling.
Enable Blink	<003>	Enable blinking for all characters.
Enable Roll	<022>	Turn on automatic scrolling.
End Blink	<017>	Write subsequent characters with blink attribute off.
End Dim	<035>	Write subsequent characters with normal brightness.
End Reverse Video	<036><105>	Write subsequent characters with normal fore/background.
End Underline	<025>	Write subsequent characters without underlines.
Enter Graphics Mode	<036><107>	Activate graphic interpreter.
Leave Graphics Mode	<036><120>	Deactivate graphic interpreter.
Model ID Report	<036><103>	Request identity report from console.
Read Cursor Position	<005>	Request cursor position report from console.
Read Pixel Value	x y VALUE	Request pixel data report from console.
Reset	<036><106> <101> or COLD	Clear entire screen, reset console to power-on state.
Set Character Background Color	<036><102> <n>	Set background color for subsequent characters.
Set Character Foreground Color	<036><101> <n>	Set foreground color for subsequent characters.
Set Cursor Position	<020><n1> <n2>	Move cursor to specified location.
Set Cursor Type	<036><106> <121><n>	Select shape of cursor.
Set Error Mode	f ERMES	Select error response for graphic commands.
Set Graphic Cursor Position	x y CURRENT	Set position of graphic cursor.
Start Blink	<016>	Write subsequent characters with blink attribute on.
Start Dim	<034>	Write subsequent characters at low intensity.
Start Replace Mode	REPLACE	Use Replace drawing mode for graphics.
Start Reverse Video	<036><104>	Write subsequent characters with reversed fore/background colors.
Start Set Mode	SET	Use Set drawing mode for graphics.
Start Underline	<024>	Write subsequent characters with underlines.
Start XOR Mode	XOR	Use XOR drawing mode for graphics.

Miscellaneous and Special Purpose Commands

Command	Form	Function
Bell	<007>	Sound the signal tone.
Define Character	<036><106> <122> <n1>... <n10>	Define a new symbol for a character.
Model ID Report	<036><103>	Request identity report from console.
Reset	<036><106> <101> or COLD	Clear entire screen, reset console to power-on state.
Set Cursor Type	<036><106> <121><n>	Select shape of cursor.
Set Error Mode	f ERMES	Select error response for graphic commands.

Command Dictionary

Attach Line

Mode:	graphic
Format:	x y LINETO
Action:	Draw a line from the current location of the graphic cursor to the specified point. The line is drawn in the current drawing color, drawing mode, and line style. After drawing, the graphic cursor is positioned at (x, y).
Related Commands:	Draw Line Set Drawing Color Set Line Style Start Replace Mode Start Set Mode Start XOR Mode
Example:	The following commands draw a five-pointed star on the screen:

```
320 390 CURRENT 14 COLOR       (position cursor, select color)
400 160 LINETO                 (draw 1st side)
170 300 LINETO                 (draw 2nd side)
450 300 LINETO                 (draw 3rd side)
240 160 LINETO                 (draw 4th side)
320 390 LINETO                 (draw last side)
```

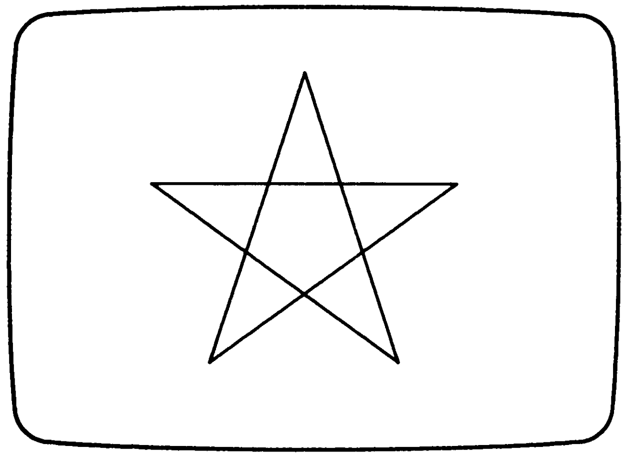


Figure 3-2 Attach line

Bell

Mode:	character
Format:	<007>
Action:	Sound the console's signal beeper.

Carriage Return

Mode:	character
Format:	<015>
Action:	Move the character cursor to the leftmost position on the current line.
Related Commands:	New Line

Clear

Mode: graphic

Format: c CLEAR

Action: Erases all character and graphic data. All pixels are set to the specified color "c", and that color is declared to be the background color. This command also does the following:

- Turns off blink, dim, reverse video, and underline modes for character output (like the End Blink, End Dim, End Reverse Video, and End Underline commands).
- Enables character blinking for the entire screen (like the Enable Blink command).
- Puts the character cursor in the home position at the upper left corner of the screen.
- Puts the graphic cursor in position (0, 0) at the lower left corner of the screen.

The monochrome display provides two colors: 0 (black) and 1 (green). However, for compatibility with the color display, this command accepts values from 0 to 15; all non-zero values are converted to 1.

Related Commands: Erase Page

Cursor Down

Mode:	character
Format:	<032>
Action:	Moves the character cursor down one character position. If the cursor is already at the bottom line of the screen, it moves to the top position in the same column.

Cursor Left

Mode:	character
Format:	<031>
Action:	Moves the character cursor one position to the left. If the cursor is already in the leftmost position, it moves to the rightmost position in the next higher line. If it is in the upper left corner, it moves to the lower right corner.

Cursor Right

Mode:	character
Format:	<030>
Action:	Moves the character cursor one position to the right. If the cursor is already in the rightmost position, it moves to the leftmost position in the next lower line. If the cursor is in the lower right corner of the screen, this command is equivalent to New Line: it either moves the cursor to the upper left corner, or rolls the screen, depending on whether roll mode is on.

Cursor Up

Mode:	character
Format:	<027>
Action:	Moves the character up one position. If the cursor is already at the top of the screen, it moves to the bottom position in the same column.

Define Character

Mode:	character
Format:	<036> <106> <122> <n0> <n1> <n2> <n3> ... <n10>
Action:	Defines the shape that the screen will display for the specified character.

The value of <n0> determines which character is to be defined. Note that characters 0 through 40_8, 177_8 through 240_8, and 377_8 are reserved, and may not be used with this command.

The characters <n1> through <n10> define the pixel pattern for the new definition, as shown in the example below. All bits that are set to 1 produce foreground or *on* pixels, and all bits that are set to 0 produce background or *off* pixels.

In general, the top two rows of pixels should be used only for upper case characters, and the bottom two rows of pixels should be used only for letters with descending tails such as lower case *p* and *q*. Also, you may want to avoid putting *on* pixels in the outermost pixel positions of the character, since this could cause adjacent characters to merge together into one shape.

Example: Suppose you are writing a mathematical program, and you want to define a character for the Greek letter *pi*. First you must draw a *pi* figure in an 8-by-10 pixel graph. Then convert this to binary numbers using 1 for each *on* pixel, and 0 for each *off* pixel. Then convert the binary numbers to decimal or octal, as shown below:

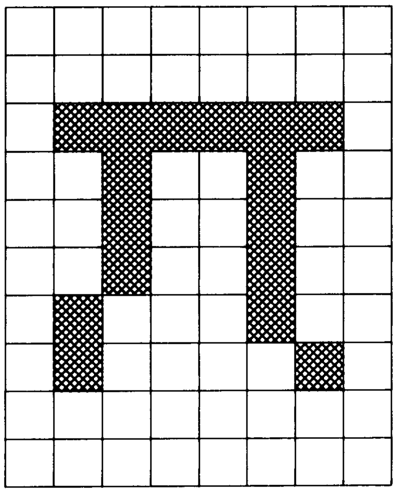
Row	Picture	Binary	Decimal	Octal
1		00000000	0	0
2		00000000	0	0
3		01111110	126	176
4		00100100	36	44
5		00100100	36	44
6		00100100	36	44
7		01000100	68	104
8		01000010	66	102
9		00000000	0	0
10		00000000	0	0

Figure 3-3 Define character

You must also decide which character to redefine. For this example we will redefine the lower case p <160>. So, the complete character sequence that we must send is:

<036> <106> <122> <160>
<000> <000> <176> <044> <044> <044> <104>
<102> <000> <000>

Define Palette

Mode:	graphic
Format:	n c p DEFPAL
Action:	This command is intended for use with color displays. For compatibility, the monchrome display accepts the command, but takes no action.

Disable Blink

Mode:	character
Format:	<004>
Initial Condition:	Blinking is enabled by power-up, Reset, Clear, or Erase Page.
Action:	Turns off the blinking feature for all characters. This command does not affect the blink attributes of the individual characters.
Related Commands:	Clear Enable Blink End Blink Erase Page Start Blink

Disable Roll

Mode:	character
Format:	<023>
Initial Condition:	Roll mode is enabled after power-up or Reset.
Action:	Turns off roll mode. After this command, if you send a New Line when the cursor is on the bottom line of the screen, the cursor will move to the top left corner of the screen.
Related Commands:	Enable Roll New Line

Draw Line

Mode:	graphic
Format:	x1 y1 x2 y2 LINE
Action:	Draws a line between the two specified points. The line is drawn using the current drawing color, drawing mode, and line style. After drawing, the graphic cursor is positioned at (x2, y2).
Related Commands:	Attach Line Draw Line Set Drawing Color Start Replace Mode Start Set Mode Start XOR Mode
Example:	The following set of commands draw a "tic-tac-toe" grid on the screen:

```
220 0 220 479 LINE
420 0 420 479 LINE
0 140 639 140 LINE
0 340 639 340 LINE
```

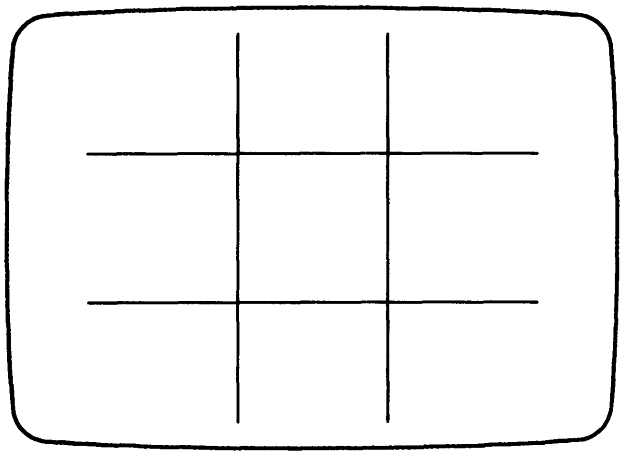


Figure 3-4 Draw line

Draw Point

Mode:	graphic
Format:	x y POINT
Action:	Draws a single point at the specified location. The pixel is set to the current drawing color.
Related Commands:	Set Drawing Color
Example:	The following command draws a single point at the center of the screen:
	320 240 POINT

Draw Text

Mode:	graphic
Format:	Any text enclosed in apostrophes (') or quotation marks (").
Action:	Draws a series of characters into the graphic plane. The text may include any character codes, but control characters are ignored. The text is drawn in the current drawing color and drawing mode.
	Up to 80 characters may be drawn at a time. The first character is drawn with its lower left corner located at the current position of the graphic cursor. After drawing, the graphic cursor will be positioned at the lower right corner of the last character.
Related Commands:	Define Character Set Drawing Color Start Replace Mode Start Set Mode Start XOR Mode
Example:	'Data General Corporation'

Enable Blink

Mode:	character
Format:	<003>
Initial Condition:	Blinking is enabled by power-up, Reset, Clear, or Erase Page.
Action:	Turns on blinking for any characters that have the blink attribute.
Related Commands:	Disable Blink End Blink Start Blink

Enable Roll

Mode:	character
Format:	<022>
Initial Condition:	Roll mode is enabled by power-up or Reset.
Action:	Turns on roll mode. After this command, if you send a New Line to the screen when the character cursor is on the bottom line of the screen, the screen rolls (as described under the New Line command).
Related Commands:	Disable Roll New Line

End Blink

Mode:	character
Format:	<017>
Initial Condition:	Blink mode is turned off by power-up, Reset, Clear, or Erase Page.
Action:	Turns off blink mode. After this command, all characters that you write on the screen will have their blink attribute turned off.
Related Commands:	Disable Blink Enable Blink Start Blink

End Dim

Mode:	character
Format:	<035>
Initial Condition:	Dim mode is turned off by power-up, Reset, Clear, or Erase Page.
Action:	Turns off dim mode. After this command, all characters that you write to the screen will have their dim attribute turned off; that is, they will be displayed at normal brightness.
Related Commands:	Start Dim

End Reverse Video

Mode:	character
Format:	<036> <105>
Initial Condition:	Reverse video mode is turned off by power-up, Reset, Clear, or Erase Page.
Action:	Turns off reverse video mode. After this command, all characters that you write to the screen will have normal foreground and background colors.
Related Commands:	Set Character Background Color Set Character Foreground Color Start Reverse Video

End Underline

Mode:	character
Format:	<025>
Initial Condition:	Underline mode is turned off by power-up, Reset, Clear, or Erase Page.
Action:	Turns off underline mode. After this command, all characters that you write to the screen will be displayed without underlines.
Related Commands:	Start Underline

Enter Graphics Mode

Mode:	character
Format:	<036> <107>
Initial Condition:	Graphics mode is turned off by power-up or Reset.
Action:	Turns on graphics mode. After this command, all characters that you send to the display will be handled by the graphic interpreter. In this mode, you may only use the graphic commands until you send a Leave Graphic Mode command.
Related Commands:	Leave Graphics Mode

Erase Page

Mode:	character
Format:	<014>
Action:	Erases the entire screen, and sets all pixels to the background color. This command also turns off the reverse video, blink, dim, and underline modes, and moves the cursor to the home position in the upper left corner of the screen.
Related Commands:	Clear Erase to End of Line Erase to End of Page

Erase to End of Line

Mode:	character
Format:	<013>
Action:	Erases all characters from the current position of the character cursor to the end of the line, and sets all pixels in that area to the background color. The cursor does not move.
Related Commands:	Clear Erase Page Erase to End of Page

Erase to End of Page

Mode:	character
Format:	<036> <106> <106>
Action:	Erases all characters from the current position of the character cursor to the end of the line, and all characters below the cursor (if any). This command sets all pixels in the affected area to the current background color. The cursor does not move.
Related Commands:	Clear Erase Page Erase to End of Line

Fill Box

Mode: graphic

Format: x1 y1 x2 y2 FILLBOX

Action: Draws a rectangular box on the screen. The two points, (x1, y1) and (x2, y2) are diagonally opposite corners of the box: either top-left and bottom-right, or top-right and bottom-left. The box is drawn using the current drawing color, drawing mode, and fill style.

Related Commands: Set Drawing Color
Set Fill Style
Start Replace Mode
Start Set Mode
Start XOR Mode

Example: These commands will clear the whole screen to black, and set the right half to solid green:

```
0 CLEAR 65535 FSTYLE 320 0 639 479 FILLBOX
```

Home Cursor

Mode: character

Format: <010>

Action: Moves the character cursor to the *home* position at the top left corner of the screen.

Leave Graphics Mode

Mode: graphic

Format: <036> <120>

Initial Condition: Graphics mode is turned off by power-up or Reset.

Action: Turns off graphic mode. After this command, all characters that you send to the display will be handled normally; they will not be processed by the graphic interpreter.

Related Commands: Enter Graphics Mode

NOTE *It is advisable to send a New Line character to the screen before using Leave Graphics Mode, to make sure that all graphic commands are executed.*

Model ID Report

Mode: character

Format: <036> <103>

Action: Instructs the terminal to send out six characters of identifying information. Your program can read these characters as though they had been typed on the keyboard by a user. The format of the characters is:

<036> <157> <043> <n1> <n2> <n3>

<n1> is <070> if your console has a monochrome display, or <071> for a color display. Values of <072>, <073>, etc. are reserved for future models.

<n2> identifies the language version of the keyboard, as follows:

Character		Language
octal	decimal	
<100>	64	reserved
<101>	65	United States
<102>	66	United Kingdom
<103>	67	French
<104>	68	German
<105>	69	Swedish/Finnish
<106>	70	Spanish
<107>	71	Danish/Norwegian
<110>	72	Italian
<111>	73	reserved
<112>	74	Swiss/German
<113>	75	reserved
<114>	76	reserved
<115>	77	reserved
<116>	78	reserved
<117>	79	reserved

<n3> identifies your D/200 emulator program. A value of <100> means no emulator; <101> means revision 1, <102> means revision 2, etc., up to <137> for revision 31.

Related Commands: Read Cursor Position
Read Pixel Value

New Line

Mode: character

Format: <012>

Action: Starts a new line of characters by moving the character cursor down one row and left to the edge of the screen. If the cursor is already at the bottom of the screen, the action taken depends on whether roll mode is on. If it is on, the following actions occur:

1. The entire screen "rolls" or shifts upward so that the top row of characters, and the upper ten rows of graphic pixels, are lost.

2. The lowest ten rows of graphic pixels are set to the current background color.

3. The character cursor is positioned at the bottom left corner of the screen.

If roll mode is off, the cursor moves to the top left corner of the screen without any rolling or erasing action.

Related Commands: Carriage Return
Disable Roll
Enable Roll

Read Cursor Position

Mode: character

Format: <005>

Action: Causes the console to send a character sequence that identifies the current position of the character cursor. Your program can read these characters as though they had been typed by a user.

The console sends your program a character sequence of the form:

<037> <n1> <n2>

<n1> is the column number (0 to 79), and <n2> is the row number (0 to 23).

Related Commands: Model ID Report
Read Pixel Value
Set Cursor Position

Read Pixel Value

Mode: graphic

Format: x y VALUE

Action: Reads the color number (0 or 1) stored in the specified pixel. The console sends the value as two characters, either 00 or 01. Your program can then read these characters as though they had been typed by a user.

This command does not change the position of the graphic cursor.

Reset

Mode:	character and graphic
Format:	<036> <106> <101> (character) COLD (graphic)
Action:	Sets the console status to the same as when the power is turned on. Specifically, this command takes the following actions:

Erases all characters, and sets all graphic pixels to 0.

Turns off blink, dim, underline, and reverse video modes for characters.

Enables screen roll and blinking.

Selects foreground color 1 and background color 0 for characters.

Selects character cursor type 2 (reverse video block), and positions the cursor at 0, 0 (upper left corner).

Turns off graphic mode and graphic blinking.

Positions the graphic cursor at 0, 0 (lower left corner).

Selects graphic drawing color 1, and background color 0.

Sets line and fill styles to 65535 (solid).

Selects Set mode for drawing.

Sets the error mode flag to 0.

Related Commands:	Clear Erase Page

Select Palette

Mode:	graphic
Format:	p SELPAL
Action:	This command is intended for use with color displays. For compatibility, the monchrome display accepts the command, but takes no action.

Set Character Background Color

Mode:	character
Format:	<036> <102> <n>
Initial Condition:	Set to 0 by power-up or Reset.
Action:	Selects background color <n> for characters. The monochrome display provides two colors: 0 (black) or 1 (green). However, for compatibility with the color display, you may use any <n> from 0 to 7; the console converts all non-zero values to 1. If <n> is greater than 7, only the lowest bits are used to select the color; thus you may use the regular digit characters, "0" <060> through "7" <067>.
Related Commands:	Set Character Foreground Color

Set Character Foreground Color

Mode:	character
Format:	<036> <101> <n>
Initial Condition:	Set to 1 by power-up or Reset.
Action:	Selects foreground color <n> for characters. The monochrome display provides two colors: 0 (black) or 1 (green). However, for compatibility with the color display, you may use any <n> from 0 to 7; the console converts all non-zero values to 1. If <n> is greater than 7, only the lowest bits are used to select the color; thus you may use the regular digit characters, "0" <060> through "7" <067>.
Related Commands:	Set Character Background Color

Set Cursor Position

Mode:	character
Format:	<020> <n1> <n2>
Initial Condition:	Set to 0, 0 (upper left corner) by power-up, Erase Page, or Reset.
Action:	Moves the character position to column <n1> of row <n2>.
Related Commands:	Read Cursor Position Set Graphic Cursor Position

Set Cursor Type

Mode: character

Format: <036> <106> <121> <n>

Initial Condition: Set to 2 by power-up or Reset.

Action: Selects the shape for the character cursor, according to the value of <n> as listed below:

0	no cursor
1	blinking underline
2	reverse video block
3	blinking reverse video block

If <n> is greater than 3, only the lowest two bits are used to select the shape; thus you may use the regular digit characters, "0" <060> through "3" <063>.

Set Drawing Color

Mode:	graphic
Format:	c COLOR
Initial Condition:	Set to 1 by power-up or Reset.
Action:	Selects the drawing color to be used for graphic drawing commands. The monochrome display provides two colors: 0 (black) and 1 (green). However, for compatibility with the color display, this command accepts values from 0 to 15; all non-zero values are converted to 1.
Related Commands:	Attach Line Draw Line Draw Point Draw Text Fill Box

Set Error Mode

Mode:	graphic
Format:	f ERMES
Initial Condition:	Set to 0 by power-up or Reset.
Action:	Selects how the console responds to errors in graphic commands. If f is non-zero, the console sends the characters <036> <115> whenever it detects an error, such as an X or Y coordinate out of range. Your program can read these characters as if they had been typed at the keyboard by a user. If f is zero, the console ignores erroneous commands.

CAUTION *If your program is not designed to handle the <036> <115> character sequence, it may cause your program to fail. In particular, command interpreters such as the Data General CLI may respond to <036> <115> by sending an error message to the screen. Since the screen is in graphic mode, the message will result in an another <036> <115>, which will produce another error message, etc. For this reason, you should use the error-reporting feature carefully.*

Set Fill Style

Mode:	graphic
Format:	n1 n2 ... n16 FSTYLE
Initial Condition:	Set to 65535 (solid fill) by power-up or Reset.
Action:	Sets the pattern of *on* and *off* pixels that is used by the Fill Box command. As many as sixteen numbers may be used to specify the pattern. If you supply less than sixteen, the console will generate the entire pattern by repeating the numbers that you supplied. Each number may range from 0 to 65535.
Related Commands:	Fill Box Set Line Style
Example:	Suppose you want to fill some boxes with a pattern of diagonal stripes. First draw the pattern as a graph, then convert it to binary numbers, using 0 for "off" pixels, and 1 for "on" pixels. Then convert the binary to decimal, as shown below:

Row	Pattern	Binary	Decimal
16	▨□□□▨□□□▨□□□▨□□□	1000100010001000	34952
15	□□□▨□□□▨□□□▨□□□▨	0001000100010001	4369
14	□□▨□□□▨□□□▨□□□▨□	0010001000100010	8738
13	□▨□□□▨□□□▨□□□▨□□	0100010001000100	17476
12	▨□□□▨□□□▨□□□▨□□□	1000100010001000	34952
11	□□□▨□□□▨□□□▨□□□▨	0001000100010001	4369
10	□□▨□□□▨□□□▨□□□▨□	0010001000100010	8738
9	□▨□□□▨□□□▨□□□▨□□	0100010001000100	17476
8	▨□□□▨□□□▨□□□▨□□□	1000100010001000	34952
7	□□□▨□□□▨□□□▨□□□▨	0001000100010001	4369
6	□□▨□□□▨□□□▨□□□▨□	0010001000100010	8738
5	□▨□□□▨□□□▨□□□▨□□	0100010001000100	17476
4	▨□□□▨□□□▨□□□▨□□□	1000100010001000	34952
3	□□□▨□□□▨□□□▨□□□▨	0001000100010001	4369
2	□□▨□□□▨□□□▨□□□▨□	0010001000100010	8738
1	□▨□□□▨□□□▨□□□▨□□	0100010001000100	17476

Figure 3-5 Set fill style

You have probably noticed that this pattern repeats once every four rows; therefore there is no need to specify all sixteen. The command

```
17476 8738 4369 34952 FSTYLE
```

will cause the striped pattern to be used for all Fill Box commands. Note that the numbers are ordered from bottom to top.

Set Graphic Blink

Mode:	graphic
Format:	f BLINK
Initial Condition:	Set to 0 by power-up or Reset.
Action:	This command is intended for use with color displays. For compatibility, the monchrome display accepts the command, but takes no action.

Set Graphic Cursor Position

Mode:	graphic
Format:	x y CURRENT
Initial Condition:	Set to 0, 0 (lower left corner) by power-up or Reset.
Action:	Positions the graphic cursor at the specified pixel. (Since the graphic cursor is not visible, this command does not change the display.)
Related Commands:	Set Cursor Position

Set Line Style

Mode:	graphic
Format:	n LSTYLE
Initial Condition:	Set to 65535 (solid lines) by power-up or Reset.
Action:	Sets the pattern of *on* and *off* pixels that is used for drawing lines. The number n may range from 0 to 65535.
Related Commands:	Attach Line Draw Line Set Fill Style
Example:	To generate a particular type of dashed or dotted line, you must first design a pixel pattern, and then convert it to a number. Suppose you want to draw a draftsman's centerline, with alternating dots and dashes. The pixel pattern might be:

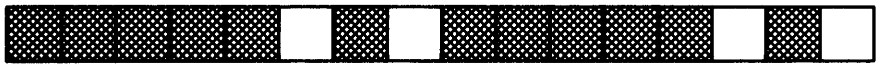


Figure 3-6 Set line style

The binary equivalent is 1111101011111010, which in decimal is 64250. Therefore you use the command:

```
64250 LSTYLE
```

Now, the Draw Line and Attach Line commands will always draw lines in this style. To return to drawing solid lines, use the command:

```
65535 LSTYLE
```

Start Blink

Mode:	character
Format:	<016>
Initial Condition:	Blink mode is turned off by power-up, Clear, Erase Page, or Reset.
Action:	Turns on blink mode. After this command, all characters written to the screen will have their blink attribute turned on.
Related Commands:	Disable Blink Enable Blink End Blink

Start Dim

Mode:	character
Format:	<034>
Initial Condition:	Dim mode is turned off by power-up, Clear, Erase Page, or Reset.
Action:	Turns on dim mode. After this command, all characters written to the screen will have their dim attribute turned on.
Related Commands:	End Dim

Start Replace Mode

Mode:	graphic
Format:	REPLACE
Initial Condition:	Turned off (Set mode is selected) by power-up or Reset.
Action:	Causes all line, box, and text drawing commands to use Replace mode. See the section on "Drawing Modes" earlier in this chapter for details.
Related Commands:	Attach Line Draw Line Draw Text Fill Box Start Replace Mode Start XOR Mode

Start Reverse Video

Mode:	character
Format:	<036> <104>
Initial Condition:	Reverse video mode is turned off by power-up, Clear, Erase Page, or Reset.
Action:	Turns on reverse video mode. After this command, all characters written to the screen will have their foreground and background colors reversed.
Related Commands:	End Reverse Video Set Background Color Set Foreground Color

Start Set Mode

Mode:	graphic
Format:	SET
Initial Condition:	Turned on by power-up or Reset.
Action:	Causes all line, box, and text drawing commands to use Set mode. See the section on "Drawing Modes" earlier in this chapter for details.
Related Commands:	Attach Line Draw Line Draw Text Fill Box Start Replace Mode Start XOR Mode

Start Underline

Mode:	character
Format:	<024>
Initial Condition:	Underline mode is turned off by power-up, Clear, Erase Page, or Reset.
Action:	Turns on underline mode. After this command, all characters written to the screen will have underlines.
Related Commands:	End Underline

Start XOR Mode

Mode:	graphic
Format:	XOR
Initial Condition:	Turned off (Set mode is selected) by power-up, Clear, Erase Page, or Reset.
Action:	Causes all line, box, and text drawing commands to use XOR mode. See the section on "Drawing Modes" earlier in this chapter for details.
Related Commands:	Attach Line Draw Line Draw Text Fill Box Start Replace Mode Start Set Mode
Example:	The following sequence of commands draws three green overlapping boxes. Because of XOR mode, all areas where two boxes overlap will be black. The area where all three boxes overlap will be green.

```
XOR                         (select XOR mode)

200 250 400 400 FILLBOX     (draw 1st box)

300 200 600 350 FILLBOX     (draw 2nd box)

100 100 500 300 FILLBOX     (draw 3rd box)
```

Color Display 4

This chapter describes the console's color display. It gives a description of its features, and a detailed reference on the various commands.

The color display is functionally divided into two separate *planes*; that is, it has separate memories for the alphanumeric (character) and graphic images. The character plane is *in front of* the graphic plane (see Figure 4-1). If you try to display characters and graphics in the same place, the characters will overwrite the picture. Either plane can be erased and rewritten without destroying data on the other plane.

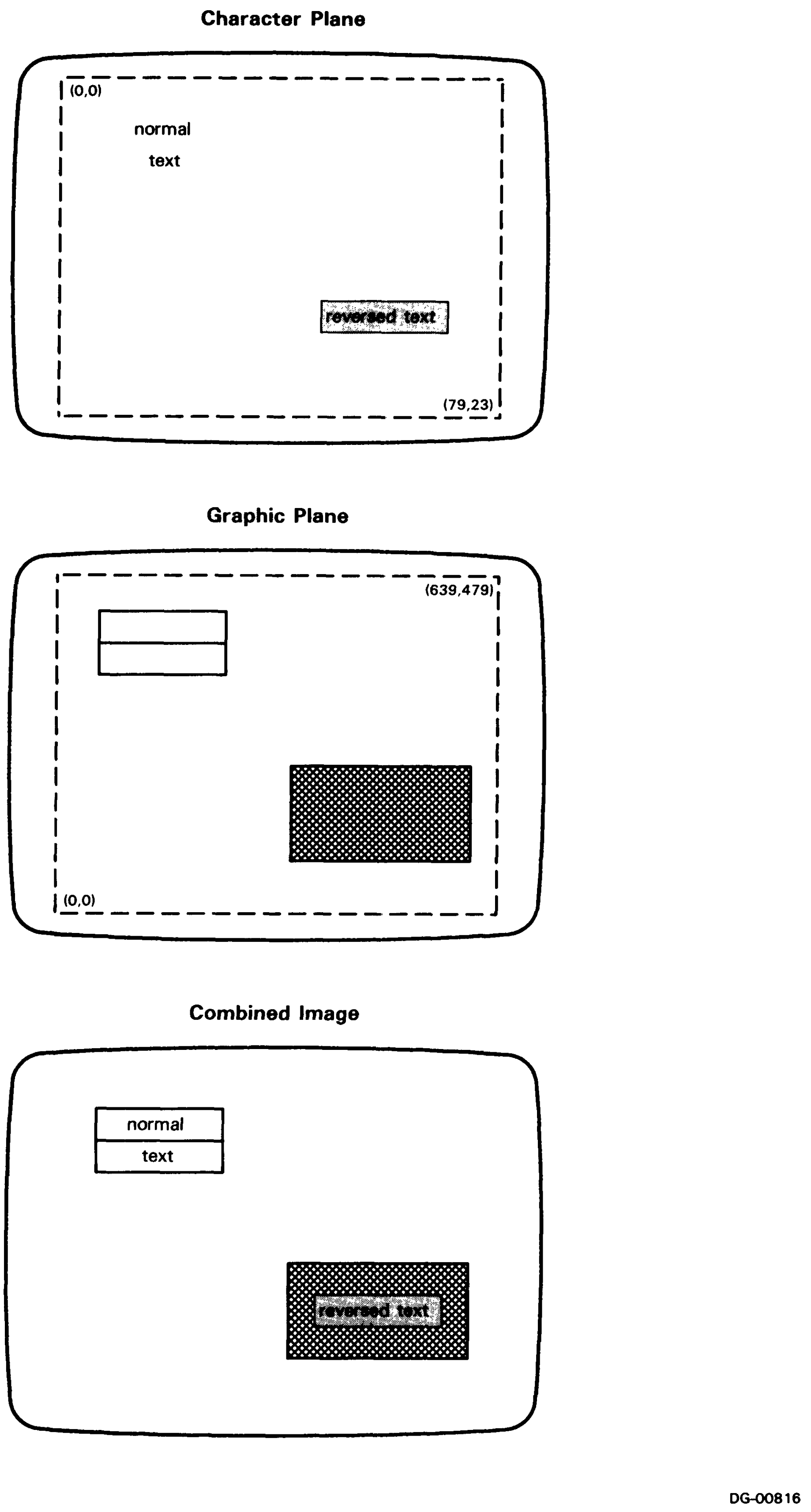


Figure 4-1 Color planes

Character Plane

The character plane contains 24 lines of 80 characters. Character positions are referenced by their X and Y coordinates, which are the column and line number. The upper left corner has coordinates (0, 0); the lower right has coordinates (79, 23).

The computer can generate 256 different characters, including the standard ASCII set, and a number of foreign language symbols. You can use the Define Character command to create your own characters, either by redefining the existing ones or by giving definitions to the unused ones.

Each character position has several *attributes* that you can control. These attributes are compatible with those provided by the IBM PC. They are:

Foreground (character) color: eight choices.

Background color: eight choices.

Video mode: normal or reversed.

Brightness: normal or dim.

Blinking: on or off.

Underline: on or off.

The computer maintains a *cursor* to mark the position at which the next character will be written. You can select several different cursor shapes, or make it invisible with the Set Cursor Type command. By sending various control characters to the display, you can move the cursor to any position on the screen. A program can also interrogate the screen, to read the cursor's current position.

Graphic Plane

The graphic plane consists of a matrix of points called *pixels*. The screen is 640 pixels wide and 240 pixels high. The computer draws pictures on the graphic plane by setting individual pixels to any of sixteen colors. Since even a simple drawing can involve hundreds or thousands of pixels, the console has built-in commands that help you to draw lines, boxes, and text. *Style* commands permit you to easily create dotted or dashed lines, and boxes with stripes or other texture patterns.

To design a graphic image, you must specify the X and Y coordinates on the screen where various objects are to be drawn. The lower left corner of the screen has coordinates (0, 0); the upper right corner has coordinates (639, 479). Note that, although the screen is physically 240 pixels high, the console firmware defines it to be 480 pixels high. If you try to write to a pixel with an odd-numbered Y coordinate, the console will actually write to the next lower even-numbered pixel. For example, an attempt to plot a point at location (345, 123) will be directed to (345, 122); the Y coordinate is converted, but the X coordinate is unchanged.

The graphic plane has a cursor that is used to write or read any pixel. Unlike the character plane cursor, the graphic cursor is not visible. However, the computer keeps track of its location for use by various graphic commands.

Color Selection

The color of a pixel is controlled by three pieces of data: the pixel value (0 to 15), the current palette (0 to 3), and the palette register value (0 to 4095). This three-level organization provides many options for creating displays that are functional as well as visually appealing. Figure 4-2 depicts the color selection process.

Pixels

Each pixel on the graphic plane is represented by four bits of the computer's memory. The four bits can store any number from 0 to 15; hence there are sixteen colors available.

One of the sixteen colors is always specified to be the *background*, and the other fifteen are *foreground*. At power-up, color 0 (black) is selected as the background. You can select a different background color with the Clear command. The choice of background color affects the operation of the graphic output commands, since they can draw in both foreground and background colors.

The graphic plane has a blinking mode that you can turn on and off with the Set Graphic Blink command. When it is on, colors 8 through 15 will alternate between their normal value and the background color. (Colors 0 through 7 are not affected.)

Palettes

To display a pixel on the screen, the console hardware takes the number from the pixel, and uses it to select one of sixteen registers in the current *palette*. Each palette register contains 12 bits, so it can have any value from 0 to 4095. This number actually determines the pixel color. Thus, although only sixteen colors may be used at one time, each of those sixteen may be chosen from a total spectrum of 4096.

This method of color selection has several advantages. It increases the variety of colors available in a low-cost system. Also, it permits rapid color changes. For instance, if color number 4 has been set to red, and you use the Define Palette command to change it to yellow, everything on the screen that is drawn in color 4 will instantly change from red to yellow.

Four separate palettes, numbered 0 to 3, are available for the graphic plane. You can change palettes at any time with the Select Palette command, which changes all sixteen colors at once.

Color Spectrum

As mentioned above, each palette register contains twelve bits that select the actual color to be displayed. There are four bits for each of the three primary colors: red, green, and blue. You can produce any color in the spectrum by mixing the three primary colors. Yellow, for example, is created by combining red and green. Brown is simply "dark yellow": red and green at a low intensity. Various shades of gray are created by mixing equal amounts of all three colors.

You use the Define Palette command to specify the value for a register. At power-up, the palettes are initialized to the values listed in tables 4-1 through 4-4. Studying these tables will give you some ideas on how to create other colors.

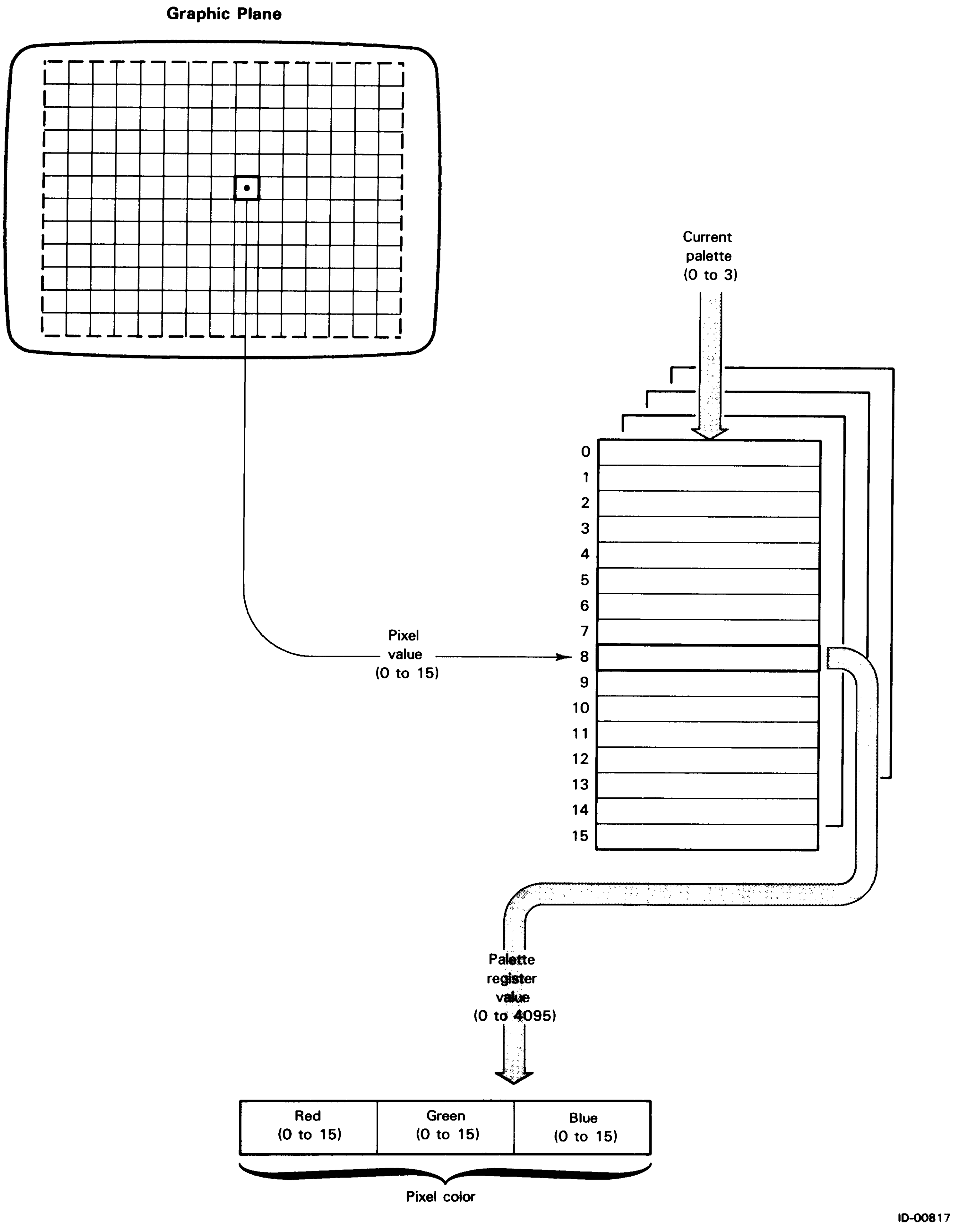


Figure 4-2 Color selection

Table 4-1 Graphic color palette 0

		Palette Register Value				
Pixel Value	Color Name	decimal red	decimal green	decimal blue	binary	decimal
0	Black	0	0	0	000000000000	0
1	Dark Blue	0	0	10	000000001010	10
2	Dark Green	0	10	0	000010100000	160
3	Dark Cyan	0	10	10	000010101010	176
4	Dark Red	10	0	0	101000000000	2560
5	Dark Magenta	10	0	10	101000001010	2576
6	Brown	10	10	0	101010100000	2720
7	Light Gray	10	10	10	101010101010	2736
8	Dark Gray	6	6	6	011001100110	1638
9	Blue	0	0	15	000000001111	15
10	Green	0	15	0	000011110000	240
11	Cyan	0	15	15	000011111111	255
12	Red	15	0	0	111100000000	3840
13	Magenta	15	0	15	111100001111	3855
14	Yellow	15	15	0	111111110000	4080
15	White	15	15	15	111111111111	4095

Table 4-2 Graphic color palette 1

		Palette Register Value				
Pixel Value	Color Name	decimal red	decimal green	decimal blue	binary	decimal
0	Black	0	0	0	000000000000	0
1	Red	15	0	0	111100000000	3840
2	Green	0	15	0	000011110000	240
3	Yellow	15	15	0	111111110000	4080
4	Blue	0	0	15	000000001111	15
5	Magenta	15	0	15	111100001111	
6	Cyan	0	15	15	000011111111	255
7	White	15	15	15	111111111111	4095
8	Black	0	0	0	000000000000	0
9	Red	15	0	0	111100000000	3840
10	Green	0	15	0	000011110000	240
11	Yellow	15	15	0	111111110000	4080
12	Blue	0	0	15	000000001111	15
13	Magenta	15	0	15	111100001111	3855
14	Cyan	0	15	15	000011111111	255
15	White	15	15	15	111111111111	4095

Table 4-3 Graphic color palette 2

		Palette Register Value				
Pixel Value	Color Name	decimal red	decimal green	decimal blue	binary	decimal
0	Black	0	0	0	000000000000	0
1	White	15	15	15	111111111111	4095
2	Red	15	0	0	111100000000	3840
3	Green	0	15	0	000011110000	240
4	Blue	0	0	15	000000001111	15
5	Cyan	0	15	15	000011111111	255
6	Magenta	15	0	15	111100001111	3855
7	Yellow	15	15	0	111111110000	4080
8	Orange	15	10	0	111110100000	4000
9	Yellow-green	10	15	0	101011110000	2800
10	Green-cyan	0	15	10	000011111010	250
11	Blue-cyan	0	10	15	000010101111	175
12	Blue-magenta	10	0	15	101000001111	2575
13	Red-magenta	15	0	10	111100001010	3850
14	Dark Gray	6	6	6	011001100110	1638
15	Light Gray	10	10	10	101010101010	2736

Table 4-4 Graphic color palette 3

Pixel Value	Color Name	Palette Register Value decimal red	green	blue	binary	decimal
0	Black	0	0	0	000000000000	0
1	White	15	15	15	111111111111	4095
2	Light Gray	14	14	14	111011101110	3822
3	.	13	13	13	110111011101	3549
4	.	12	12	12	110011001100	3276
5	.	11	11	11	101110111011	3003
6	.	10	10	10	101010101010	2730
7	(progressive	9	9	9	100110011001	2457
8	shades	8	8	8	100010001000	2184
9	of	7	7	7	011101110111	1911
10	gray)	6	6	6	011001100110	1638
11	.	5	5	5	010101010101	1365
12	.	4	4	4	010001000100	1092
13	.	3	3	3	001100110011	819
14	.	2	2	2	001000100010	546
15	Dark Gray	1	1	1	000100010001	273

Drawing Functions

Lines and Points

The console provides two commands for line drawing. To use the Draw Line command, you specify the start and end points of the line. To use the Attach Line command, you specify only the end point; the starting point is the current position of the graphic cursor. The Attach Line command is convenient when you are drawing a complex shape with a series of connected line segments.

To draw dotted or dashed lines, you use the Set Line Style command to specify a pattern of *on* and *off* pixels. When you draw a line, the *on* and *off* pixels are set to different colors. The specific colors that are used depend on the current drawing color, background color, and on the drawing mode as described below.

The console provides a Draw Point command to change the color of a single pixel. This is equivalent to drawing a line with its start and end point at the same location.

Boxes

The console provides a Fill Box command for drawing rectangular boxes of any size. Boxes may be drawn in a solid color, or in a texture pattern such as stripes or checks.

To draw textured boxes, you use the Set Box Style command to specify a pattern of *on* and *off* pixels. When you draw a box, the *on* and *off* pixels are set to different colors. The specific colors that are used depend on the current drawing color, background color, and on the drawing mode as described below.

Text

For maximum freedom in combining text and pictures, the console has a Draw Text command that lets you write characters into the graphic plane. Text in the graphic plane has several features that are not available in the character plane. Characters are drawn in the colors of the graphic palette, so you can select from 4096 possible colors. Also, characters can be positioned anywhere on the screen, unlike the character plane where characters must fit exactly into the 80-by-24 grid.

To draw a character, the console reads the pattern of *on* and *off* pixels that is stored in the character set memory. This is the same memory that is used for the character plane. You can define new character shapes, or redefine old ones, with the Define Character command.

When you draw a character, the *on* and *off* pixels are set to different colors. The specific colors that are used depend on the current drawing color, background color, and on the drawing mode as described below.

Drawing Modes

The console's Start Drawing Mode commands provide three different ways of drawing the *on* and *off* pixels that are specified by the line style, box style, or character shape. The three modes are called Set, Replace, and XOR (pronounced "ex-or"). In Set mode, which is selected at power-up, the *off* pixels are left in their previous color. For example, if you draw a dashed line across a picture, the spaces between the dashes will be unchanged.

In Replace mode, the *off* pixels are set to the current background color. For example, if you draw a dashed line across a picture, the spaces between the dashes will be set to the background color. The background color is the one that was last used in a Clear command. At power-up, color 0 (black) is selected as the background.

In XOR mode, the *off* pixels are left unchanged. The *on* pixels, however, are set to a color which is chosen by taking the exclusive-OR of the drawing color and the current pixel value. The result is that, when you try to draw one color over another, a third color is actually produced.

The exclusive-or function produces a result in which each bit of one number is complemented if the corresponding bit of the other number is 1. So, for example, if you draw with color 3 (binary 0011) onto a pixel containing color 5 (binary 0101), the result will be color 6 (binary 0110).

XOR mode produces some odd effects, but it has the unique advantage that you can "undraw" any part of a picture by simply drawing it twice. Continuing with our example above, if we draw with color 3 on the pixel that now contains 6, the result will be the original color, 5. This effect is most useful for programs that display temporary items, such as a menu or cursor, in front of a picture. The temporary material can be quickly removed, and your program will not need to laboriously redraw the original picture.

Note that XOR mode works differently with color 0, since XORing with 0, like adding 0, does not change a number. Therefore, drawing with color 0 has no effect, and drawing any color onto pixels containing 0 produces the unmodified drawing color. However, the principle of "undrawing" by drawing twice still applies.

Programming

The simplest way to control the display is by simply sending characters to it. In addition to the "printing" characters (letters, digits, and other symbols), there are a number of characters that have special control functions. Some of these control characters, such as NEW LINE and ERASE PAGE, are common to most terminals that use the ASCII character set. Other characters have functions such as moving the cursor, changing colors or attributes, and defining new characters. Of particular importance is the Enter Graphic Mode character sequence, since it changes the way the display handles other characters, and provides easy access to the graphic functions.

When using the console, you should remember that the character and graphic data are in two separate planes, and commands that affect one may not affect the other. For instance, if you change the colors of some of the characters, it will not affect the colors in the graphic plane.

In general, character commands do not work in graphic mode, and vice versa. If your program manipulates both planes, you must remember to enter and leave graphic mode, depending on which type of command you want to use next.

Interpreter Mode

The graphic interpreter is a firmware program that is part of every Desktop Generation computer. Its function is to read text commands such as POINT, LINE, etc., and perform the specified functions. This makes it very easy for you to create pictures in any programming language, including command languages such as the Data General CLI.

To activate the interpreter, you send an Enter Graphic Mode character sequence to the display. When the console is in graphic mode, all characters that you send to it are interpreted as graphic commands. The commands consist of keywords and numbers. The numbers that specify the parameters for a command must be sent *before* the keyword. For example, to select color 6 for drawing, send the text

```
6 COLOR
```

to the display. To draw a line from the lower left corner of the screen (coordinates 0, 0) to the upper right corner (coordinates 639, 479), send

```
0 0 639 479 LINE
```

When you finish drawing a picture, you may send a Leave Graphic Mode character sequence to the display. This deactivates the graphic interpreter, so that all characters that you send to the display will be written to the character plane in the normal manner.

In the above examples, the numbers and words are separated by spaces. In fact you can use one or more of the space, comma, or NEW LINE characters as separators between parts of a command.

When you send commands to the interpreter, it stores them in a memory buffer. A command is not actually executed until either the buffer becomes full, or you send a NEW LINE character. Therefore, to ensure that all commands are executed, you should always send a NEW LINE to the screen before leaving graphics mode.

Although the above examples use only upper case letters for commands, the console accepts either upper or lower case.

Assembler Language Access

For advanced programs, or those that must run at high speed, you can program the display in assembler language. This eliminates the need for the graphic interpreter to read and interpret text commands. Instead, your program can jump directly into the computer's firmware by using the EHYP instruction. Assembler language access is described in detail in Appendix C. More detailed information on the computer may be found in the *Model 10 and 10/SP Computer Systems Reference*, DGC no. 014-000766.

Command Summary

The following conventions are used for describing display command formats:

- Numbers enclosed in angle brackets represent the character codes that you send to the display to perform a function. For example, Erase Page is defined as <014>, so to erase the screen, you must send a control-L (14 octal, 12 decimal) to the display.
- Letters enclosed in angle brackets represent character codes whose values depend on the application. For more information, consult the Reference section later in this chapter.
- Upper case sans serif words are text that should be sent to the display exactly as shown. For example, to select color 4 for drawing in the graphic plane, you would send the text 4 COLOR. (Note that you may actually use either upper or lower case in your programs; this manual uses only upper case for clarity.)
- Lower case letters represent numbers that may have a range of values. Specifically:

  x, x1, and x2 represent horizontal coordinates in the graphic plane, and may range from 0 to 639.

  y, y1, and y2 represent vertical coordinates in the graphic plane, and may range from 0 to 479.

  c represents a pixel color value, and may range from 0 to 15.

  f represents a number used as a logical flag. A value of zero means False or Off, and any other value means True or On.

  p represents a palette number, and may range from 0 to 3.

  n, n1, n2, etc., represent arbitrary numbers whose value depends on the specific application. For more information, consult the Reference section later in this chapter.

Graphic Output Commands

Command	Form	Function
Attach Line	x y LINETO	Draw a line from current position to specified point.
Clear	c CLEAR	Clear screen and select background color.
Define Palette	n c p DEFPAL	Select color value for palette register.
Draw Line	x1 y1 x2 y2 LINE	Draw a line between the specified points.
Draw Point	x y POINT	Draw a single point.
Draw Text	'text' or "text"	Write characters on the graphic plane.
Erase Page	<014>	Erase entire screen and reset cursors and character modes.
Erase to End of Line	<013>	Erase screen from cursor position to end of line.
Erase to End of Page	<036><106> <106>	Erase screen from cursor position to end of page.
Fill Box	x1 y1 x2 y2 FILLBOX	Draw and fill a rectangular box at the specified location.
Reset	<036><106> <101> or COLD	Clear entire screen, reset console to power-on state.
Select Palette	p SELPAL	Select graphic color palette.
Set Drawing Color	c COLOR	Select color for graphic commands.
Set Fill Style	n1 ... n16 FSTYLE	Specify texture pattern for filling boxes.
Set Graphic Blink	f BLINK	Turn graphic blinking on or off.
Set Graphic Cursor Position	x y CURRENT	Set position of graphic cursor.
Set Line Style	n LSTYLE	Set pattern for line drawing (solid, dotted, dashed, etc.)
Start Replace Mode	REPLACE	Use Replace drawing mode for graphics.
Start Set Mode	SET	Use Set drawing mode for graphics.
Start XOR Mode	XOR	Use XOR drawing mode for graphics.

Character Output Commands

Command	Form	Function
Define Character	<036><106> <122> <n1>... <n10>	Define a new symbol for a character.
End Blink	<017>	Write subsequent characters with blink attribute off.
End Dim	<035>	Write subsequent characters with normal brightness.
End Reverse Video	<036><105>	Write subsequent characters with normal fore/background.
End Underline	<025>	Write subsequent characters without underlines.
Erase Page	<014>	Erase entire screen and reset cursors and character modes.
Erase to End of Line	<013>	Erase screen from cursor position to end of line.
Erase to End of Page	<036><106> <106>	Erase screen from cursor position to end of page.
Reset	<036><106> <101> or COLD	Clear entire screen, reset console to power-on state.
Set Character Background Color	<036><102> <n>	Set background color for subsequent characters.
Set Character Foreground Color	<036><101> <n>	Set foreground color for subsequent characters.
Start Blink	<016>	Write subsequent characters with blink attribute on.
Start Dim	<034>	Write subsequent characters at low intensity.
Start Reverse Video	<036><104>	Write subsequent characters with reversed fore/background colors.
Start Underline	<024>	Write subsequent characters with underlines.

Cursor Control Commands

Command	Form	Function
Carriage Return	<015>	Return cursor to left margin.
Cursor Down	<032>	Move cursor down one position.
Cursor Left	<031>	Move cursor left one position.
Cursor Right	<030>	Move cursor right one position.
Cursor Up	<027>	Move cursor up one position.
Enter Graphics Mode	<036><107>	Activate graphic interpreter.
Erase Page	<014>	Erase entire screen and move cursor to upper left corner.
Home Cursor	<010>	Move cursor to top left corner of screen.
Leave Graphics Mode	<036><120>	Deactivate graphic interpreter.
New Line	<012>	Move cursor to start of next line.
Read Cursor Position	<005>	Request cursor position report from console.
Reset	<036><106> <101> or COLD	Clear entire screen, reset console to power-on state.
Set Cursor Position	<020><n1> <n2>	Move character cursor to specified location.
Set Cursor Type	<036><106> <121><n>	Select shape of cursor.
Set Graphic Cursor Position	x y CURRENT	Move graphic cursor to specified location.

Status and Control Commands

Command	Form	Function
Disable Blink	<004>	Disable blinking for the whole screen.
Disable Roll	<023>	Turn off automatic scrolling.
Enable Blink	<003>	Enable blinking for all characters.
Enable Roll	<022>	Turn on automatic scrolling.
End Blink	<017>	Write subsequent characters with blink attribute off.
End Dim	<035>	Write subsequent characters with normal brightness.
End Reverse Video	<036><105>	Write subsequent characters with normal fore/background.
End Underline	<025>	Write subsequent characters without underlines.
Enter Graphics Mode	<036><107>	Activate graphic interpreter.
Leave Graphics Mode	<036><120>	Deactivate graphic interpreter.
Model ID Report	<036><103>	Request identity report from console.
Read Cursor Position	<005>	Request cursor position report from console.
Read Pixel Value	x y VALUE	Request pixel data report from console.
Reset	<036><106> <101> or COLD	Clear entire screen, reset console to power-on state.
Set Character Background Color	<036><102> <n>	Set background color for subsequent characters.
Set Character Foreground Color	<036><101> <n>	Set foreground color for subsequent characters.
Set Cursor Position	<020><n1> <n2>	Move cursor to specified location.
Set Cursor Type	<036><106> <121><n>	Select shape of cursor.
Set Error Mode	f ERMES	Select error response for graphic commands.
Set Graphic Cursor Position	x y CURRENT	Set position of graphic cursor.
Start Blink	<016>	Write subsequent characters with blink attribute on.
Start Dim	<034>	Write subsequent characters at low intensity.
Start Replace Mode	REPLACE	Use Replace drawing mode for graphics.
Start Reverse Video	<036><104>	Write subsequent characters with reversed fore/background colors.
Start Set Mode	SET	Use Set drawing mode for graphics.
Start Underline	<024>	Write subsequent characters with underlines.
Start XOR Mode	XOR	Use XOR drawing mode for graphics.

Miscellaneous and Special Purpose Commands

Command	Form	Function
Bell	<007>	Sound the signal tone.
Define Character	<036><106> <122> <n1>... <n10>	Define a new symbol for a character.
Model ID Report	<036><103>	Request identity report from console.
Reset	<036><106> <101> or COLD	Clear entire screen, reset console to power-on state.
Set Cursor Type	<036><106> <121><n>	Select shape of cursor.
Set Error Mode	f ERMES	Select error response for graphic commands.

Command Dictionary

Attach Line

Mode:	graphic
Format:	x y LINETO
Action:	Draw a line from the current location of the graphic cursor to the specified point. The line is drawn in the current drawing color, drawing mode, and line style. After drawing, the graphic cursor is positioned at (x, y).
Related Commands:	Define Palette Draw Line Set Drawing Color Set Line Style Start Replace Mode Start Set Mode Start XOR Mode

Example The following commands draw a five-pointed star on the screen:

```
320 390 CURRENT 14 COLOR      (position cursor, select color)
400 160 LINETO                (draw 1st side)
170 300 LINETO                (draw 2nd side)
450 300 LINETO                (draw 3rd side)
240 160 LINETO                (draw 4th side)
320 390 LINETO                (draw last side)
```

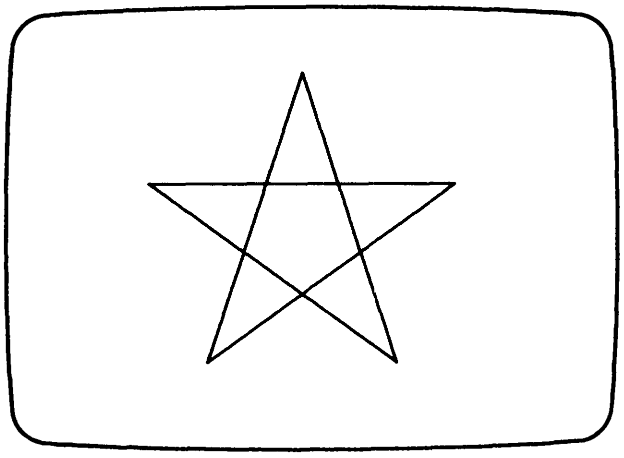


Figure 4-3 Attach line

Bell

Mode:	character
Format:	<007>
Action:	Sound the console's signal beeper.

Carriage Return

Mode:	character
Format:	<015>
Action:	Move the character cursor to the leftmost position on the current line.
Related Commands:	New Line

Clear

Mode: graphic

Format: c CLEAR

Action: Erases all character and graphic planes. All pixels in the graphic plane are set to the specified color c, and that color is declared to be the background color. This command also does the following:

- Turns off blink, dim, reverse video, and underline modes for character output (like the End Blink, End Dim, End Reverse Video, and End Underline commands).
- Enables character blinking for the entire screen (like the Enable Blink command).
- Puts the character cursor in the home position at the upper left corner of the screen.
- Puts the graphic cursor in position (0, 0) at the lower left corner of the screen.

Related Commands: Erase Page

Cursor Down

Mode:	character
Format:	<032>
Action:	Moves the character cursor down one character position. If the cursor is already at the bottom line of the screen, it moves to the top position in the same column.

Cursor Left

Mode:	character
Format:	<031>
Action:	Moves the character cursor one position to the left. If the cursor is already in the leftmost position, it moves to the rightmost position in the same line.

Cursor Right

Mode:	character
Format:	<030>
Action:	Moves the character cursor one position to the right. If the cursor is already in the rightmost position, it moves to the leftmost position in the same line.

Cursor Up

Mode:	character
Format:	<027>
Action:	Moves the character up one position. If the cursor is already at the top of the screen, it moves to the bottom position in the same column.

Define Character

Mode: character

Format: <036> <106> <122> <n0>
<n1> <n2> <n3> ... <n10>

Action: Defines the shape that the screen will display for the specified character.

The value of <n0> determines which character is to be defined. Note that characters 0 through 40_8, 177_8 through 240_8, and 377_8 are reserved, and may not be used with this command.

The characters <n1> through <n10> define the pixel pattern for the new definition, as shown in the example below. All bits that are set to 1 produce foreground or *on* pixels, and all bits that are set to 0 produce background or *off* pixels.

In general, the top two rows of pixels should be used only for upper case characters, and the bottom two rows of pixels should be used only for letters with descending tails such as lower case *p* and *q*. Also, you may want to avoid putting *on* pixels in the outermost pixel positions of the character, since this could cause adjacent characters to merge together into one shape.

Example: Suppose you are writing a mathematical program, and you want to define a character for the Greek letter *pi*. First you must draw a *pi* figure in an 8-by-10 pixel graph. Then convert this to binary numbers using 1 for each *on* pixel, and 0 for each *off* pixel. Then convert the binary numbers to decimal or octal, as shown below:

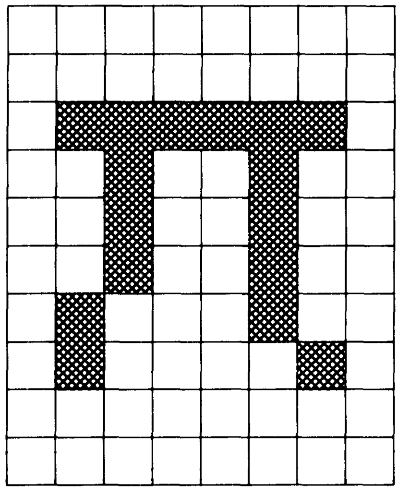
Row	Picture	Binary	Decimal	Octal
1		00000000	0	0
2		00000000	0	0
3		01111110	126	176
4		00100100	36	44
5		00100100	36	44
6		00100100	36	44
7		01000100	68	104
8		01000010	66	102
9		00000000	0	0
10		00000000	0	0

Figure 4-4 Define character

You must also decide which character to redefine. For this example we will redefine the lower case p <160>. So, the complete character sequence that we must send is:

<036> <106> <122> <160>
<000> <000> <176> <044> <044> <044> <104>
<102> <000> <000>

Define Palette

Mode:	graphic
Format:	n c p DEFPAL
Action:	Defines a new color value for a palette register. In the command, n is the new register value (0 to 4095), c is the register number (0 to 15), and p is the palette number (0 to 3).
Related Commands:	Select Palette Set Drawing Color
Example:	Suppose you want to set color 5 in palette 1 to light blue. The desired color value is 8 red, 8 green, and 15 blue. Convert this to a single number using the formula: (red * 256) + (green * 16) + blue = 2191 So you send the following command: 2191 5 1 DEFPAL

Disable Blink

Mode:	character
Format:	<004>
Initial Condition:	Blinking is enabled by power-up, Reset, Clear, or Erase Page.
Action:	Turns off the blinking feature for all characters. This command does not affect the blink attributes of the individual characters.
Related Commands:	Clear Enable Blink End Blink Erase Page Set Graphic Blink Start Blink

Disable Roll

Mode:	character
Format:	<023>
Initial Condition:	Roll mode is enabled after power-up or Reset.
Action:	Turns off roll mode. After this command, if you send a New Line when the cursor is on the bottom line of the screen, the cursor will move to the top left corner of the screen.
Related Commands:	Enable Roll New Line

Draw Line

Mode: graphic

Format: `x1 y1 x2 y2 LINE`

Action: Draws a line between the two specified points. The line is drawn using the current drawing color, drawing mode, and line style. After drawing, the graphic cursor is positioned at (x2, y2).

Related Commands: Attach Line

Define Palette
Draw Line
Set Drawing Color
Start Replace Mode
Start Set Mode
Start XOR Mode

Example: The following set of commands draw a "tic-tac-toe" grid on the screen:

```
220 0 220 479 LINE
420 0 420 479 LINE
0 140 639 140 LINE
0 340 639 340 LINE
```

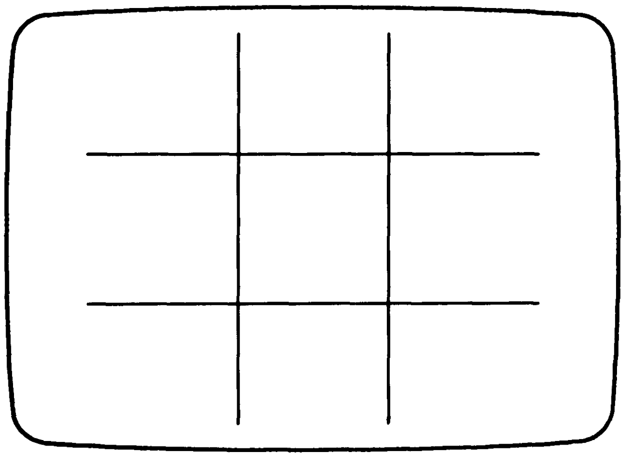


Figure 4-5 Draw line

Draw Point

Mode:	graphic
Format:	x y POINT
Action:	Draws a single point at the specified location. The pixel is set to the current drawing color.
Related Commands:	Set Drawing Color
Example:	The following command draws a single point at the center of the screen: 320 240 POINT

Draw Text

Mode:	graphic
Format:	Any text enclosed in apostrophes (') or quotation marks (").
Action:	Draws a series of characters into the graphic plane. The text may include any character codes, but control characters are ignored. The text is drawn in the current drawing color and drawing mode. Up to 80 characters may be drawn at a time. The first character is drawn with its lower left corner located at the current position of the graphic cursor. After drawing, the graphic cursor will be positioned at the lower right corner of the last character.
Related Commands:	Define Character Set Drawing Color Start Replace Mode Start Set Mode Start XOR Mode
Example:	The following commands draw three words of text in three different colors: 11 COLOR ' Data ' 12 COLOR ' General ' 13 COLOR 'Corporation '

Enable Blink

Mode:	character
Format:	<003>
Initial Condition:	Blinking is enabled by power-up, Reset, Clear, or Erase Page.
Action:	Turns on blinking for any characters that have the blink attribute.
Related Commands:	Disable Blink End Blink Set Graphic Blink Start Blink

Enable Roll

Mode:	character
Format:	<022>
Initial Condition:	Roll mode is enabled by power-up or Reset.
Action:	Turns on roll mode. After this command, if you send a New Line to the screen when the character cursor is on the bottom line of the screen, the screen rolls (as described under the New Line command).
Related Commands:	Disable Roll New Line

End Blink

Mode:	character
Format:	<017>
Initial Condition:	Blink mode is turned off by power-up, Reset, Clear, or Erase Page.
Action:	Turns off blink mode. After this command, all characters that you write on the screen will have their blink attribute turned off.
Related Commands:	Disable Blink Enable Blink Set Graphic Blink Start Blink

End Dim

Mode:	character
Format:	<035>
Initial Condition:	Dim mode turned off by power-up, Reset, Clear, or Erase Page.
Action:	Turns off dim mode. After this command, all characters that you write to the screen will have their dim attribute turned off; that is, they will be displayed at normal brightness.
Related Commands:	Start Dim

End Reverse Video

Mode:	character
Format:	<036> <105>
Initial Condition:	Reverse video mode is turned off by power-up, Reset, Clear, or Erase Page.
Action:	Turns off reverse video mode. After this command, all characters that you write to the screen will have normal foreground and background colors.
Related Commands:	Set Character Background Color Set Character Foreground Color Start Reverse Video

End Underline

Mode:	character
Format:	<025>
Initial Condition:	Underline mode is turned off by power-up, Reset, Clear, or Erase Page.
Action:	Turns off underline mode. After this command, all characters that you write to the screen will be displayed without underlines.
Related Commands:	Start Underline

Enter Graphics Mode

Mode:	character
Format:	<036> <107>
Initial Condition:	Graphics mode is turned off by power-up or Reset.
Action:	Turns on graphics mode. After this command, all characters that you send to the display will be handled by the graphic interpreter. In this mode, you may only use the graphic commands until you send a Leave Graphic Mode command.
Related Commands:	Leave Graphics Mode

Erase Page

Mode:	character
Format:	<014>
Action:	Erases the entire screen, and sets all pixels to the background color. This command also turns off the reverse video, blink, dim, and underline modes.
Related Commands:	Clear Erase to End of Line Erase to End of Page

Erase to End of Line

Mode:	character
Format:	<013>
Action:	Erases all characters from the current position of the character cursor to the end of the line, and sets all pixels in that area to the background color.
Related Commands:	Clear Erase Page Erase to End of Page

Erase to End of Page

Mode:	character
Format:	<036> <106> <106>
Action:	Erases all characters from the current positon of the character cursor to the end of the line, and also all characters below the cursor (if any). This command sets all pixels in the affected area to the current background color.
Related Commands:	Clear Erase Page Erase to End of Line

Fill Box

Mode:	graphic
Format:	x1 y1 x2 y2 FILLBOX
Action:	Draws a rectangular box on the screen. The two points, (x1, y1) and (x2, y2) are diagonally opposite corners of the box: either top-left and bottom-right, or top-right and bottom-left. The box is drawn using the current drawing color, drawing mode. and fill style.
Related Commands:	Set Drawing Color Set Fill Style Start Replace Mode Start Set Mode Start XOR Mode
Example:	This set of commands draws three boxes in different colors:

```
4 COLOR 100 100 200 200 FILLBOX
5 COLOR 200 200 300 300 FILLBOX
6 COLOR 300 300 400 400 FILLBOX
```

Home Cursor

Mode:	character
Format:	<010>
Action:	Moves the character cursor to the *home* position at the top left corner of the screen.

Leave Graphics Mode

Mode:	graphic
Format:	<036> <120>
Initial Condition:	Graphics mode is turned off by power-up or Reset.
Action:	Turns off graphic mode. After this command, all characters that you send to the display will be handled normally; they will not be processed by the graphic interpreter.
Related Commands:	Enter Graphics Mode

NOTE *It is advisable to send a New Line character to the screen before using Leave Graphics Mode, to make sure that all graphic commands are executed.*

	Generated Codes		
Key Pressed	**decimal**	**octal**	**hex**
CTRL-F15	30 48	036 060	1E 30
CTRL-F1	30 49	036 061	1E 31
CTRL-F2	30 50	036 062	1E 32
CTRL-F3	30 51	036 063	1E 33
CTRL-F4	30 52	036 064	1E 34
CTRL-F5	30 53	036 065	1E 35
CTRL-F6	30 54	036 066	1E 36
CTRL-F7	30 55	036 067	1E 37
CTRL-F8	30 56	036 070	1E 38
CTRL-F9	30 57	036 071	1E 39
CTRL-F10	30 58	036 072	1E 3A
CTRL-F11	30 59	036 073	1E 3B
CTRL-F12	30 60	036 074	1E 3C
CTRL-F13	30 61	036 075	1E 3D
CTRL-F14	30 62	036 076	1E 3E
SHIFT-C1 CTRL-SHIFT-C1	30 88	036 130	1E 58
SHIFT-C2 CTRL-SHIFT-C2	30 89	036 131	1E 59
SHIFT-C3 CTRL-SHIFT-C3	30 90	036 132	1E 5A
SHIFT-C4 CTRL-SHIFT-C4	30 91	036 133	1E 5B
C1 CTRL-C1	30 92	036 134	1E 5C
C2 CTRL-C2	30 93	036 135	1E 5D
C3 CTRL-C3	30 94	036 136	1E 5E
C4 CTRL-C4	30 95	036 137	1E 5F

	Generated Codes		
Key Pressed	decimal	octal	hex
SHIFT-F15	30 96	036 140	1E 60
SHIFT-F1	30 97	036 141	1E 61
SHIFT-F2	30 98	036 142	1E 62
SHIFT-F3	30 99	036 143	1E 63
SHIFT-F4	30 100	036 144	1E 64
SHIFT-F5	30 101	036 145	1E 65
SHIFT-F6	30 102	036 146	1E 66
SHIFT-F7	30 103	036 147	1E 67
SHIFT-F8	30 104	036 150	1E 68
SHIFT-F9	30 105	036 151	1E 69
SHIFT-F10	30 106	036 152	1E 6A
SHIFT-F11	30 107	036 153	1E 6B
SHIFT-F12	30 108	036 154	1E 6C
SHIFT-F13	30 109	036 155	1E 6D
SHIFT-F14	30 110	036 156	1E 6E
F15	30 112	036 160	1E 70
F1	30 113	036 161	1E 71
F2	30 114	036 162	1E 72
F3	30 115	036 163	1E 73
F4	30 116	036 164	1E 74
F5	30 117	036 165	1E 75
F6	30 118	036 166	1E 76
F7	30 119	036 167	1E 77
F8	30 120	036 170	1E 78
F9	30 121	036 171	1E 79
F10	30 122	036 172	1E 7A
F11	30 123	036 173	1E 7B
F12	30 124	036 174	1E 7C
F13	30 125	036 175	1E 7D
F14	30 126	036 176	1E 7E

Programming Examples B

This Appendix contains a sample program that makes use of many of the system console's graphics features. Figure B-1 shows the display that the program produces. The program can run on either a monochrome or color display.

Figure B-2 is a program listing in MP/BASIC, which is compatible with most BASIC languages. Figure B-3 is a listing of an equivalent program in SP/Pascal.

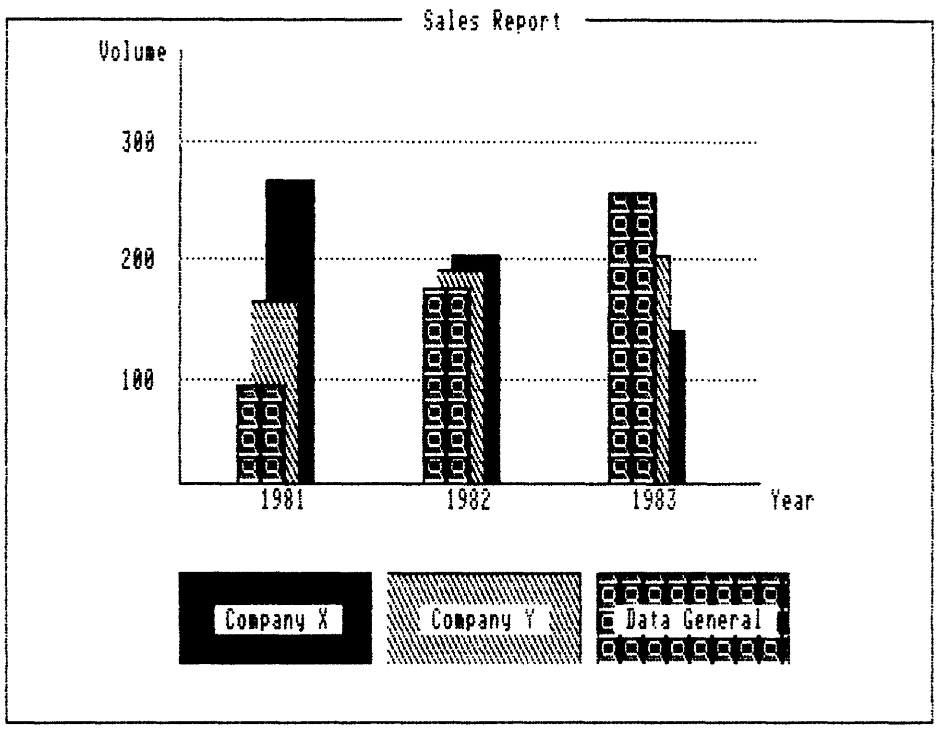


Figure B-1

```
500 REM =====================================================
510 REM              Sample Bar Chart Program
520 REM         for DESKTOP GENERATION Model 10 and 10/SP
900 REM =====================================================
910 REM ======================== initialize screen & write text labels
1000 PRINT CHR$(30);CHR$(71);" COLD "
1020 PRINT CHR$(16);CHR$(35);CHR$(0);
1030 PRINT CHR$(30);CHR$(68);" Sales Report "
1035 PRINT CHR$(28);
1040 PRINT CHR$(16);CHR$(7);CHR$(1);" Volume "
1041 PRINT CHR$(16);CHR$(65);CHR$(16);" Year "
1048 PRINT CHR$(30);CHR$(69);
1050 PRINT CHR$(16);CHR$(22);CHR$(16);"1981"
1051 PRINT CHR$(16);CHR$(38);CHR$(16);"1982"
1052 PRINT CHR$(16);CHR$(54);CHR$(16);"1983"
1060 PRINT CHR$(16);CHR$(10);CHR$(12);"100"
1061 PRINT CHR$(16);CHR$(10);CHR$(8);"200"
1062 PRINT CHR$(16);CHR$(10);CHR$(4);"300"
1070 PRINT CHR$(29)
1080 REM ====================== draw borders & dotted lines
1090 PRINT CHR$(30);CHR$(71);
1095 PRINT " REPLACE "
1100 PRINT "15 COLOR 120 450 120 160 LINE 520 160 LINETO "
1150 PRINT "272 470 0 470 LINE"
1155 PRINT "0 0 LINETO 639 0 LINETO"
1160 PRINT "639 470 LINETO 400 470 LINETO"
1200 PRINT "17476 LSTYLE "
1210 FOR Y=230 TO 390 STEP 80
1220   PRINT 120;Y;520;Y;" LINE"
1230 NEXT Y
1299 PRINT "65535 LSTYLE "
1400 REM ====================== draw legend boxes & graph bars
1500 FOR I=0 TO 2
1505   PRINT (2^I)+8;" COLOR "
1510   GOSUB 4000
1520 NEXT I
1600 PRINT "7 COLOR 144 60 CURRENT"
1610 PRINT " ' Company X ' "
1620 PRINT "286 60 CURRENT ' Company Y ' "
1630 PRINT "420 60 CURRENT ' Data General ' "
1700 GOTO 9000
3900 REM =============== Main box drawing routine:
3990 REM ====================== draw legend box
4000 READ N
4010 FOR J=1 TO N
4020   READ K
4030   PRINT K;
4040 NEXT J
4050 PRINT " FSTYLE "
5000 LET X1=(144*I)+120
5010 LET Y1=40
5020 LET X2=X1+132
5030 LET Y2=100
5040 PRINT X1;Y1;X2;Y2;" FILLBOX "
5045 PRINT " 14 COLOR "
5050 PRINT X2;Y1;X2;Y2;" LINE ";X1;Y2;" LINETO "
```

Figure B-2

Model ID Report

Mode: character

Format: <036> <103>

Action: Instructs the terminal to send out six characters of identifying information. Your program can read these characters as though they had been typed on the keyboard by a user. The format of the characters is:

<036> <157> <043> <n1> <n2> <n3>

<n1> is <070> if your console has a monochrome display, or <071> for a color display.

<n2> identifies the language version of the keyboard, as follows:

Character		Language
octal	**decimal**	
<100>	64	reserved
<101>	65	United States
<102>	66	United Kingdom
<103>	67	French
<104>	68	German
<105>	69	Swedish/Finnish
<106>	70	Spanish
<107>	71	Danish/Norwegian
<110>	72	Italian
<111>	73	reserved
<112>	74	Swiss/German
<113>	75	reserved
<114>	76	reserved
<115>	77	reserved
<116>	78	reserved
<117>	79	reserved

<n3> identifies the revision number of your console. A value of <100> means revision 0, <101> means revision 1, etc., up to <137> for revision 31.

Related Commands: Read Cursor Position
Read Pixel Value

New Line

Mode:	character
Format:	<012>

Action:

Starts a new line of characters by moving the character cursor down one row and left to the edge of the screen. If the cursor is already at the bottom of the screen, the action taken depends on whether roll mode is on. If it is on, the following actions occur:

1. The entire screen "rolls" or shifts upward so that the top row of characters, and the upper ten rows of graphic pixels, are lost.

2. The lowest ten rows of graphic pixels are set to the current background color.

3. The character cursor is positioned at the bottom left corner of the screen.

If roll mode is off, the cursor moves to the top left corner of the screen without any rolling or erasing action.

Related Commands:

Carriage Return
Disable Roll
Enable Roll

Read Cursor Position

Mode:	character
Format:	<005>

Action:

Causes the console to send a character sequence that identifies the current position of the character cursor. Your program can read these characters as though they had been typed by a user.

The console sends your program a character sequence of the form:

<037> <n1> <n2>

<n1> is the column number (0 to 79), and <n2> is the row number (0 to 23).

Related Commands:

Model ID Report
Read Pixel Value
Set Cursor Position

Read Pixel Value

Mode:	graphic
Format:	x y VALUE
Action:	Reads the color number (0 to 15) stored in the specified pixel. The console sends the value as two characters, which your program can then read as though they had been typed by a user. For example, if the pixel color is 12, the console sends 12. If the pixel color, is 5, the console sends 05. This command does not change the position of the graphic cursor.

Reset

Mode:	character and graphic
Format:	<036> <106> <101> (character) COLD (graphic)
Action:	Sets the console status to the same as when the power is turned on. Specifically, this command takes the following actions: Erases all characters, and sets all graphic pixels to 0. Turns off blink, dim, underline, and reverse video modes for characters. Enables screen roll and blinking. Selects foreground color 2 and background color 0 for characters. Selects character cursor type 2 (reverse video block), and positions the cursor at 0, 0 (upper left corner). Turns off graphic mode and graphic blinking. Positions the graphic cursor at 0, 0 (lower left corner). Selects graphic drawing color 2, and background color 0. Sets line and fill styles to 65535 (solid). Selects Set mode for drawing. Sets the error mode flag to 0.
Related Commands:	Clear Erase Page

Select Palette

Mode:	graphic
Format:	p SELPAL
Action:	Selects the specified color palette (0 to 3). All pixels in the graphic plane will be displayed in the colors of the selected palette; thus this command can change all sixteen colors at once.
Related Commands:	Define Palette

Set Character Background Color

Mode:	character
Format:	<036> <102> <n>
Initial Condition:	Set to 0 by power-up or Reset.
Action:	Selects background color <n> for characters. <n> may range from 0 to 7. If <n> is greater than 7, only the lowest three bits are used to select the color; thus you may use the regular digit characters, "0" <060> through "7" <067>.

The available colors are:

0	"transparent" (lets graphic plane show through)
1	dark blue
2	dark green
3	dark cyan
4	dark red
5	dark magenta
6	brown
7	gray

Related Commands: Set Character Foreground Color

Set Character Foreground Color

Mode:	character
Format:	<036> <101> <n>
Initial Condition:	Set to 1 by power-up or Reset.
Action:	Selects foreground color <n> for characters. <n> may range from 0 to 7. If <n> is greater than 7, only the lowest three bits are used to select the color; thus you may use the regular digit characters, "0" <060> through "7" <067>. Note that the actual color depends on whether a character is displayed in dim or normal mode.

The available colors are:

<n>	**Normal color**	**Dim color**
0	black	transparent (lets graphic plane show through)
1	blue	dark blue
2	green	dark green
3	cyan	dark cyan
4	red	dark red
5	magenta	dark magenta
6	yellow	brown
7	white	gray

Related Commands: Set Character Background Color

Set Cursor Position

Mode:	character
Format:	<020> <n1> <n2>
Initial Condition:	Set to 0, 0 (upper left corner) by power-up, Erase Page, or Reset.
Action:	Moves the character position to column <n1> of row <n2>.
Related Commands:	Read Cursor Position Set Graphic Cursor Position

Set Cursor Type

Mode:	character
Format:	<036> <106> <121> <n>
Initial Condition:	Set to 20 by power-up or Reset.
Action:	Selects the shape for the character cursor, according to the value of <n> as listed below:

0 no cursor

1 blinking underline

2 reverse video block

3 blinking reverse video block

If <n> is greater than 3, only the lowest two bits are used to select the shape; thus you may use the regular digit characters, "0" <060> through "3" <063>.

Set Drawing Color

Mode:	graphic
Format:	c COLOR
Initial Condition:	Set to 2 by power-up or Reset.
Action:	Selects the drawing color to be used for graphic drawing commands.
Related Commands:	Attach Line Define Palette Draw Line Draw Point Draw Text Fill Box Select Palette

Set Error Mode

Mode:	graphic
Format:	f ERMES
Initial Condition:	Set to 0 by power-up or Reset.
Action:	Selects how the console responds to errors in graphic commands. If f is non-zero, the console sends the characters <036> <115> whenever it detects an error, such as an X or Y coordinate out of range. Your program can read these characters as if they had been typed at the keyboard by a user. If f is zero, the console ignores erroneous commands.

CAUTION *If your program is not designed to handle the <036> <115> character sequence, it may cause your program to fail. In particular, command interpreters such as the Data General CLI may respond to <036> <115> by sending an error message to the screen. Since the screen is in graphic mode, the message will result in an another <036> <115>, which will produce another error message, etc. For this reason, you should use the error-reporting feature carefully.*

Set Fill Style

Mode:	graphic
Format:	n1 n2 ... n16 FSTYLE
Initial Condition:	Set to 65535 (solid fill) by power-up or Reset.
Action:	Sets the pattern of *on* and *off* pixels that is used by the Fill Box command. As many as sixteen numbers may be used to specify the pattern. If you supply less than sixteen, the console will generate the entire pattern by repeating the numbers that you supplied. Each number may range from 0 to 65535.
Related Commands:	Fill Box Set Line Style
Example:	Suppose you want to fill some boxes with a pattern of diagonal stripes. First draw the pattern as a graph, then convert it to binary numbers, and then to decimal:

Row	Pattern	Binary	Decimal
16	▨□□□▨□□□▨□□□▨□□□	1000100010001000	34952
15	□□□▨□□□▨□□□▨□□□▨	0001000100010001	4369
14	□□▨□□□▨□□□▨□□□▨□	0010001000100010	8738
13	□▨□□□▨□□□▨□□□▨□□	0100010001000100	17476
12	▨□□□▨□□□▨□□□▨□□□	1000100010001000	34952
11	□□□▨□□□▨□□□▨□□□▨	0001000100010001	4369
10	□□▨□□□▨□□□▨□□□▨□	0010001000100010	8738
9	□▨□□□▨□□□▨□□□▨□□	0100010001000100	17476
8	▨□□□▨□□□▨□□□▨□□□	1000100010001000	34952
7	□□□▨□□□▨□□□▨□□□▨	0001000100010001	4369
6	□□▨□□□▨□□□▨□□□▨□	0010001000100010	8738
5	□▨□□□▨□□□▨□□□▨□□	0100010001000100	17476
4	▨□□□▨□□□▨□□□▨□□□	1000100010001000	34952
3	□□□▨□□□▨□□□▨□□□▨	0001000100010001	4369
2	□□▨□□□▨□□□▨□□□▨□	0010001000100010	8738
1	□▨□□□▨□□□▨□□□▨□□	0100010001000100	17476

Figure 4-6 Set fill style

You have probably noticed that this pattern repeats once every four rows; therefore there is no need to specify all sixteen. The command

```
17476 8738 4369 34952 FSTYLE
```

will cause the striped pattern to be used for all Fill Box commands. Note that the numbers are ordered from bottom to top.

Set Graphic Blink

Mode:	graphic
Format:	f BLINK
Initial Condition:	Set to 0 by power-up or Reset.
Action:	Turns graphic blinking on (if f is non-zero) and off (if f is zero). When graphic blinking is on, colors 8 through 15 in the graphic plane will alternate between their normal value and the background color. (This command has no effect on character blinking.)
Related Commands:	Disable Blink Enable Blink End Blink Start Blink

Set Graphic Cursor Position

Mode:	graphic
Format:	x y CURRENT
Initial Condition:	Set to 0, 0 (lower left corner) by power-up or Reset.
Action:	Positions the graphic cursor at the specified pixel. (Since the graphic cursor is not visible, this command does not change the display.)
Related Commands:	Set Cursor Position

Set Line Style

Mode:	graphic
Format:	n LSTYLE
Initial Condition:	Set to 65535 (solid lines) by power-up or Reset.
Action:	Sets the pattern of *on* and *off* pixels that is used for drawing lines. The number n may range from 0 to 65535.
Related Commands:	Attach Line Draw Line Set Fill Style
Example:	To generate a particular type of dashed or dotted line, you must first design a pixel pattern, and then convert it to a number. Suppose you want to draw a draftsman's centerline, with alternating dots and dashes. The pixel pattern might be:

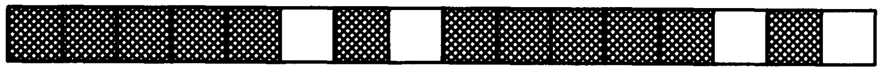


Figure 4-7 Set line style

The binary equivalent is 1111101011111010, which in decimal is 64250. Therefore you use the command:

64250 LSTYLE

Now, the Draw Line and Attach Line commands will always draw lines in this style. To return to drawing solid lines, use the command:

65535 LSTYLE

Start Blink

Mode:	character
Format:	<016>
Initial Condition:	Blink mode is turned off by power-up, Clear, Erase Page, or Reset.
Action:	Turns on blink mode. After this command, all characters written to the screen will have their blink attribute turned on.
Related Commands:	Disable Blink Enable Blink End Blink

Start Dim

Mode:	character
Format:	<034>
Initial Condition:	Dim mode is turned off by power-up, Clear, Erase Page, or Reset.
Action:	Turns on dim mode. After this command, all characters written to the screen will have their dim attribute turned on.
Related Commands:	End Dim

Start Replace Mode

Mode:	graphic
Format:	REPLACE
Initial Condition:	Turned off (Set mode is selected) by power-up or Reset.
Action:	Causes all line, box, and text drawing commands to use Replace mode. See the section on "Drawing Modes" earlier in this chapter for details.
Related Commands:	Attach Line Draw Line Draw Text Fill Box Start Replace Mode Start XOR Mode

Start Reverse Video

Mode:	character
Format:	<036> <104>
Initial Condition:	Reverse video mode is turned off by power-up, Clear, Erase Page, or Reset.
Action:	Turns on reverse video mode. After this command, all characters written to the screen will have their foreground and background colors reversed.
Related Commands:	End Reverse Video Set Background Color Set Foreground Color

Start Set Mode

Mode:	graphic
Format:	SET
Initial Condition:	Turned on by power-up or Reset.
Action:	Causes all line, box, and text drawing commands to use Set mode. See the section on "Drawing Modes" earlier in this chapter for details.
Related Commands:	Attach Line Draw Line Draw Text Fill Box Start Replace Mode Start XOR Mode

Start Underline

Mode:	character
Format:	<024>
Initial Condition:	Underline mode is turned off by power-up, Clear, Erase Page, or Reset.
Action:	Turns on underline mode. After this command, all characters written to the screen will have underlines.
Related Commands:	End Underline

Start XOR Mode

Mode:	graphic
Format:	XOR
Initial Condition:	Turned off (Set mode is selected) by power-up, Clear, Erase Page, or Reset.
Action:	Causes all line, box, and text drawing commands to use XOR mode. See the section on "Drawing Modes" earlier in this chapter for details.
Related Commands:	Attach Line Draw Line Draw Text Fill Box Start Replace Mode Start Set Mode
Example:	The following sequence of commands draws three boxes in different colors. Because of XOR mode, a total of seven colors will be produced.

```
XOR                                  (select XOR mode)
1 COLOR    200 250 400 400 FILLBOX   (draw blue box)
2 COLOR    300 200 600 350 FILLBOX   (draw green box)
4 COLOR    100 100 500 300 FILLBOX   (draw red box)
```

Character Codes A

This Appendix lists all the character codes used by the DESKTOP GENERATION system.

The following table lists the keyboard character codes and their functions. This table includes the foreign language symbols and other characters that are not actually part of the ASCII standard, but that are used on DESKTOP GENERATION Model 10 and 10/SP computers.

This Appendix is based on the U.S. version of the keyboard. Other versions may have slight differences.

NOTE *All references in this Appendix to the* up arrow *key apply to the one on the screen management keypad. (Some versions of the keyboard have another up-arrow symbol on the main keypad.)*

	Codes			
decimal	**octal**	**hex**	**Main keyboard**	**Display function**
0	000	00	CTRL-SHIFT-2 CTRL-@	Null
1	001	01	CTRL-A CTRL-SHIFT-A	
2	002	02	CTRL-B CTRL-SHIFT-B	Start reverse video
3	003	03	CTRL-C CTRL-SHIFT-C	Enable blink
4	004	04	CTRL-D CTRL-SHIFT-D	Disable blink
5	005	05	CTRL-E CTRL-SHIFT-E	Read cursor position
6	006	06	CTRL-F CTRL-SHIFT-F	
7	007	07	CTRL-G CTRL-SHIFT-G	Bell
8	010	08	CTRL-H CTRL-SHIFT-H HOME CTRL-HOME SHIFT-HOME CTRL-SHIFT-HOME	Home cursor
9	011	09	CTRL-I CTRL-SHIFT-I TAB SHIFT-TAB CTRL-TAB CTRL-SHIFT-TAB	Tab
10	012	0A	CTRL-J CTRL-SHIFT-J NEW LINE CTRL-NEW LINE SHIFT-NEW LINE CTRL-SHIFT-NEW LINE	New line

Codes				
decimal	**octal**	**hex**	**Main keyboard**	**Display function**
11	013	0B	CTRL-K CTRL-SHIFT-K ERASE EOL SHIFT-ERASE EOL CTRL-ERASE EOL CTRL-SHIFT-ERASE EOL	Erase to end of line
12	014	0C	CTRL-L CTRL-SHIFT-L ERASE PAGE SHIFT-ERASE PAGE CTRL-ERASE PAGE CTRL-SHIFT-ERASE PAGE	Erase page
13	015	0D	CTRL-M CTRL-SHIFT-M CR CTRL-CR SHIFT-CR CTRL-SHIFT-CR	Carriage return
14	016	0E	CTRL-N CTRL-SHIFT-N	Start blink
15	017	0F	CTRL-O CTRL-SHIFT-O	End blink
16	020	10	CTRL-P CTRL-SHIFT-P	Set cursor position
17	021	11	CTRL-Q CTRL-SHIFT-Q	
18	022	12	CTRL-R CTRL-SHIFT-R	Enable roll
19	023	13	CTRL-S CTRL-SHIFT-S	Disable roll
20	024	14	CTRL-T CTRL-SHIFT-T	Start underscore
21	025	15	CTRL-U CTRL-SHIFT-U	End underscore
22	026	16	CTRL-V CTRL-SHIFT-V	End reverse video
23	027	17	CTRL-W CTRL-SHIFT-W up arrow CTRL-up arrow	Cursor up

Codes				
decimal	**octal**	**hex**	**Main keyboard**	**Display function**
24	030	18	CTRL-X CTRL-SHIFT-X right arrow CTRL-right arrow	Cursor right
25	031	19	CTRL-Y CTRL-SHIFT-Y left arrow CTRL-left arrow	Cursor left
26	032	1A	CTRL-Z CTRL-SHIFT-Z down arrow CTRL-down arrow	Cursor down
27	033	1B	BREAK/ESC SHIFT-BREAK/ESC CTRL-BREAK/ESC CTRL-SHIFT- BREAK/ESC CTRL-[	Escape
28	034	1C	CTRL-\	Start dim
29	035	1D	CTRL-]	End dim
30	036	1E	CTRL-SHIFT-6	Function header
31	037	1F	CTRL-SHIFT- -	Response to Read Cursor Position command
32	040	20	Space bar SHIFT-space CTRL-space CTRL-SHIFT-space	Space
33	041	21	SHIFT-1 CTRL-SHIFT-1	! (exclamation mark)
34	042	22	SHIFT-' CTRL-SHIFT-'	" (quotation mark)
35	043	23	SHIFT-3 CTRL-SHIFT-3	# (number sign)
36	044	24	SHIFT-4 CTRL-SHIFT-4	$ (dollar sign)
37	045	25	SHIFT-5 CTRL-SHIFT-5	% (percent)
38	046	26	SHIFT-7 CTRL-SHIFT-7	& (ampersand)
39	047	27	' CTRL-'	' (closing single quotation mark or apostrophe)
40	050	28	SHIFT-9 CTRL-SHIFT-9	((opening parenthesis)

Codes				
decimal	octal	hex	Main keyboard	Display function
41	051	29	SHIFT-0 CTRL-SHIFT-0	) (Closing parenthesis)
42	052	2A	SHIFT-8 CTRL-SHIFT-8	* (asterisk)
43	053	2B	SHIFT-= CTRL-SHIFT-=	+ (plus)
44	054	2C	, CTRL-,	, (comma)
45	055	2D	- CTRL- -	- (hyphen or minus)
46	056	2E	. CTRL-.	. (period or decimal point)
47	057	2F	/ CTRL-/	/ (slash)
48	060	30	0 CTRL-0	0
49	061	31	1 CTRL-1	1
50	062	32	2 CTRL-2	2
51	063	33	3 CTRL-3	3
52	064	34	4 CTRL-4	4
53	065	35	5 CTRL-5	5
54	066	36	6 CTRL-6	6
55	067	37	7 CTRL-7	7
56	070	38	8 CTRL-8	8
57	071	39	9 CTRL-9	9
58	072	3A	SHIFT-; CTRL-SHIFT-;	: (colon)
59	073	3B	; CTRL-;	; (semicolon)

Codes				
decimal	octal	hex	Main keyboard	Display function
60	074	3C	SHIFT-, CTRL-SHIFT-,	< (less than)
61	075	3D	= CTRL-=	= (equals)
62	076	3E	SHIFT-. CTRL-SHIFT-.	> (greater than)
63	077	3F	SHIFT-/ CTRL-SHIFT-/	? (question mark)
64	100	40	SHIFT-2	@ ("at" sign)
65	101	41	SHIFT-A	A
66	102	42	SHIFT-B	B
67	103	43	SHIFT-C	C
68	104	44	SHIFT-D	D
69	105	45	SHIFT-E	E
70	106	46	SHIFT-F	F
71	107	47	SHIFT-G	G
72	110	48	SHIFT-H	H
73	111	49	SHIFT-I	I
74	112	4A	SHIFT-J	J
75	113	4B	SHIFT-K	K
76	114	4C	SHIFT-L	L
77	115	4D	SHIFT-M	M
78	116	4E	SHIFT-N	N
79	117	4F	SHIFT-O	O
80	120	50	SHIFT-P	P
81	121	51	SHIFT-Q	Q
82	122	52	SHIFT-R	R
83	123	53	SHIFT-S	S
84	124	54	SHIFT-T	T
85	125	55	SHIFT-U	U
86	126	56	SHIFT-V	V
87	127	57	SHIFT-W	W
88	130	58	SHIFT-X	X
89	131	59	SHIFT-Y	Y
90	132	5A	SHIFT-Z	Z
91	133	5B	[	[(opening bracket)

	Codes			
decimal	octal	hex	Main keyboard	Display function
92	134	5C	\	\ (reverse slant)
93	135	5D	]	] (closing bracket)
94	136	5E	SHIFT-6	^ (circumflex)
95	137	5F	SHIFT- -	_ (underline)
96	140	60	` CTRL-`	` (opening single quotation mark)
97	141	61	A	a
98	142	62	B	b
99	143	63	C	c
100	144	64	D	d
101	145	65	E	e
102	146	66	F	f
103	147	67	G	g
104	150	68	H	h
105	151	69	I	i
106	152	6A	J	j
107	153	6B	K	k
108	154	6C	L	l
109	155	6D	M	m
110	156	6E	N	n
111	157	6F	O	o
112	160	70	P	p
113	161	71	Q	q
114	162	72	R	r
115	163	73	S	s
116	164	74	T	t
117	165	75	U	u
118	166	76	V	v
119	167	77	W	w
120	170	78	X	x
121	171	79	Y	y
122	172	7A	Z	z

Codes				
decimal	octal	hex	Main keyboard	Display function
123	173	7B	SHIFT-[CTRL-SHIFT-[	{ (open brace)
124	174	7C	SHIFT-\\ CTRL-SHIFT-\\	| (vertical line)
125	175	7D	SHIFT-] CTRL-SHIFT-]	} (closing brace)
126	176	7E	SHIFT-\` CTRL-SHIFT-\`	~ (tilde)
127	177	7F	DEL CTRL-DEL SHIFT-DEL CTRL-SHIFT-DEL	Delete

International Symbols

When using this table, you should be familiar with the uses of the modifying keys, particularly SHIFT, CTRL, and SPCL, as described in Chapter 2. Remember that SHIFT and CTRL are pressed together with another key, but SPCL is pressed by itself. For example, the European "ae" character is described in this table as SPCL-A-SHIFT-E. This means that you generate this character by typing SPCL, then lower case A, and finally upper case E.

Codes				
decimal	**octal**	**hex**	**Main keyboard**	**Display function**
160	240	A0		Space
161	241	A1		Space
162	242	A2		Space
163	243	A3		Space
164	244	A4		Space
165	245	A5		Space
166	246	A6	SPCL-O-X	¤
167	247	A7	SPCL-C-/	¢
168	250	A8	SPCL-L-minus	£
169	251	A9		Space
170	252	AA		Space
171	253	AB	SPCL-SHIFT-1	¡
172	254	AC	SPCL-SHIFT-/	¿
173	255	AD		Space
	⋮			
185	271	B9		Space
186	272	BA	SPCL-grave-space	`
187	273	BB	SPCL-paragraph sign	§
188	274	BC	SPCL-degree-space	°
189	275	BD	SPCL-umlaut-space	¨
190	276	BE	SPCL-acute-space	´
191	277	BF	SPCL-3-space	↑
192	300	C0	SPCL-acute-SHIFT-A	Á
193	301	C1	SPCL-grave-SHIFT-A	À
194	302	C2	SPCL-circumflex-SHIFT-A	Â
195	303	C3	SPCL-umlaut-SHIFT-A	Ä
196	304	C4	SPCL-tilde-SHIFT-A	Ã
197	305	C5	SPCL-degree-SHIFT-A	Å
198	306	C6	SPCL-A-SHIFT-E	Æ
199	307	C7	SPCL-comma-SHIFT-C	Ç

	Codes			
decimal	**octal**	**hex**	**Main keyboard**	**Display function**
200	310	C8	SPCL-acute-SHIFT-E	É
201	311	C9	SPCL-grave-SHIFT-E	È
202	312	CA	SPCL-circumflex-SHIFT-E	Ê
203	313	CB	SPCL-umlaut-SHIFT-E	Ë
204	314	CC	SPCL-acute-SHIFT-I	Í
205	315	CD	SPCL-grave-SHIFT-I	Ì
206	316	CE	SPCL-circumflex-SHIFT-I	Î
207	317	CF	SPCL-umlaut-SHIFT-I	Ï
208	320	D0	SPCL-tilde-SHIFT-N	Ñ
209	321	D1	SPCL-acute-SHIFT-O	Ó
210	322	D2	SPCL-grave-SHIFT-O	Ò
211	323	D3	SPCL-circumflex-SHIFT-O	Ô
212	324	D4	SPCL-umlaut-SHIFT-O	Ö
213	325	D5	SPCL-tilde-SHIFT-O	Õ
214	326	D6	SPCL-slash-SHIFT-O	Ø
215	327	D7	SPCL-E-SHIFT-O	Œ
216	330	D8	SPCL-acute-SHIFT-U	Ú
217	331	D9	SPCL-grave-SHIFT-U	Ù
218	332	DA	SPCL-circumflex-SHIFT-U	Û
219	333	DB	SPCL-umlaut-SHIFT-U	Ü
220	334	DC		Space
	.			
	.			
	.			
	.			
	.			
223	337	DF		Space
224	340	E0	SPCL-acute-A	á
225	341	E1	SPCL-grave-A	à
226	342	E2	SPCL-circumflex-A	â
227	343	E3	SPCL-umlaut-A	ä
228	344	E4	SPCL-tilde-A	ã
229	345	E5	SPCL-degree-A	å
230	346	E6	SPCL-E-A	æ
231	347	E7	SPCL-comma-C	ç

Codes				
decimal	octal	hex	Main keyboard	Display function
232	350	E8	SPCL-acute-E	é
233	351	E9	SPCL-grave-E	è
234	352	EA	SPCL-circumflex-E	ê
235	353	EB	SPCL-umlaut-E	ë
236	354	EC	SPCL-acute-I	í
237	355	ED	SPCL-grave-I	ì
238	356	EE	SPCL-circumflex-I	î
239	357	EF	SPCL-umlaut-I	ï
240	360	F0	SPCL-tilde-N	ñ
241	361	F1	SPCL-acute-O	ó
242	362	F2	SPCL-grave-O	ò
243	363	F3	SPCL-circumflex-O	ô
244	364	F4	SPCL-umlaut-O	ö
245	365	F5	SPCL-tilde-O	õ
246	366	F6	SPCL-slash-O	ø
247	367	F7	SPCL-E-O	œ
248	370	F8	SPCL-acute-U	ú
249	371	F9	SPCL-grave-U	ù
250	372	FA	SPCL-circumflex-U	û
251	373	FB	SPCL-umlaut-U	ü
252	374	FC	SPCL-S-S	ß
253	375	FD		Space
254	376	FE		Space
255	377	FF		Space

Function Keys

This table lists the sequences of codes that are generated by the user-definable function keys.

	Generated Codes		
Key Pressed	**decimal**	**octal**	**hex**
CMD-SHIFT-PRINT CMD-CTRL-SHIFT-PRINT	30 1	036 001	1E 01
CMD-PRINT CMD-CTRL-PRINT	30 17	036 021	1E 11
SHIFT-up arrow CTRL-SHIFT-up arrow	30 23	036 027	1E 17
SHIFT-right arrow CTRL-SHIFT-right arrow	30 24	036 030	1E 18
SHIFT-left arrow CTRL-SHIFT-left arrow	30 25	036 031	1E 19
SHIFT-down arrow CTRL-SHIFT-down arrow	30 26	036 032	1E 1A
CTRL-SHIFT-F15	30 32	036 040	1E 20
CTRL-SHIFT-F1	30 33	036 041	1E 21
CTRL-SHIFT-F2	30 34	036 042	1E 22
CTRL-SHIFT-F3	30 35	036 043	1E 23
CTRL-SHIFT-F4	30 36	036 044	1E 24
CTRL-SHIFT-F5	30 37	036 045	1E 25
CTRL-SHIFT-F6	30 38	036 046	1E 26
CTRL-SHIFT-F7	30 39	036 047	1E 27
CTRL-SHIFT-F8	30 40	036 050	1E 28
CTRL-SHIFT-F9	30 41	036 051	1E 29
CTRL-SHIFT-F10	30 42	036 052	1E 2A
CTRL-SHIFT-F11	30 43	036 053	1E 2B
CTRL-SHIFT-F12	30 44	036 054	1E 2C
CTRL-SHIFT-F13	30 45	036 055	1E 2D
CTRL-SHIFT-F14	30 46	036 056	1E 2E

```
5052 REM ======================= draw graph bars
5055 PRINT (2^I)+8;" COLOR "
5100 FOR J=0 TO 2
5110   READ Y
5120   LET X1=(160+(128*J))+(10*(2-I))
5130   LET X2=X1+32
5140   LET Y1=162
5150   LET Y2=Y+Y1
5160   PRINT X1;Y1;X2;Y2;" FILLBOX"
5170   PRINT " 14 COLOR "
5172   PRINT X2;Y1;X2;Y2;" LINE ";X1;Y2;" LINETO "
5180   PRINT (2^I)+8;" COLOR "
5200 NEXT J
6000 RETURN
7990 REM ======================= data for fill styles & bar heights
8000 DATA 1,65535
8100 DATA 200,150,100
8109 DATA 4,17476,8738,4369,34952
8110 DATA 120,140,150
8118 DATA 9,65535,61455,59367,59367,59367,61455,60071,54611,32769
8120 DATA 64,128,192
9000 PRINT CHR$(30);CHR$(80)
```

Figure B-2

```
SP/Pascal     Rev. 1.10          Tuesday  November 1, 1983        3:10:16 PM
Compiling EXAMPLE.PAS


PROGRAM example (input, output);

        CONST
                reset_screen            = '<036><106><101>';
                start_reverse_video     = '<036><104>';
                end_reverse_video       = '<036><105>';
                start_dim               = '<034>';
                end_dim                 = '<035>';
                enter_graphic_mode      = '<036><107>';
                leave_graphic_mode      = '<036><120>';


        VAR
                i, x, y:        INTEGER;



        PROCEDURE position_text (x, y: INTEGER; t: STRING 80);

              BEGIN
                write (CHR (16), CHR (x), CHR (y), t);

              END { position_text } ;


        PROCEDURE highlight_edges (x1, y1, x2, y2: INTEGER);

              BEGIN
                writeln ('14 COLOR ',
                         x2 : 4, y1 : 4, x2 : 4, y2 : 4, ' LINE');
                writeln (x1 : 4, y2 : 4, ' LINETO');

              END { highlight_edges } ;


        PROCEDURE draw_boxes (i: INTEGER);

                TYPE
                        ia2     = ARRAY [0 .. 2] OF INTEGER;
                        sa      = ARRAY [0 .. 2] OF STRING 128;
                        chart   = ARRAY [0 .. 2] OF ia2;

                CONST
                        box_color       = ia2 [9, 10, 12];
                        box_style       = sa ['65535',
                                            '17476 8738 4369 34952',
        '65535 61455 59367 59367 59367 61455 60071 54611 32769'];
                        chart_data      = chart [
                                                ia2 [200, 150, 100],
                                                ia2 [120, 140, 150],
                                                ia2 [64, 128, 192] ];


                VAR
                        x1, y1, x2, y2: INTEGER;
                        j:              INTEGER;
```

Figure B-3

```


       BEGIN
        { select color and fill style }

        writeln (box_color [i], ' COLOR ',
                 box_style [i], ' FSTYLE');


        { draw legend box }

        x1 := (144 * i) + 120;
        y1 := 40;
        x2 := x1 + 132;
        y2 := 100;
        writeln (x1 : 4, y1 : 4, x2 : 4, y2 : 4, ' FILLBOX');
        highlight_edges (x1, y1, x2, y2);


        { draw graph bars }

        FOR j := 0 to 2 DO
           BEGIN
            writeln (box_color [i], ' COLOR');
            x1 := 160 + (128 * j) + (10 * (2 - i));
            y1 := 162;
            x2 := x1 + 32;
            y2 := y1 + chart_data [i, j];
            writeln (x1 : 4, y1 : 4, x2 : 4, y2 : 4,
                 ' FILLBOX');
            highlight_edges (x1, y1, x2, y2);
           END;

       END { draw_boxes } ;


{ --------------- MAIN PROGRAM --------------- }


   BEGIN
    { initialize screen, write labels }

    write (reset_screen, start_reverse_video);
    position_text (35, 0, ' Sales Report ');
    write (start_dim);
    position_text (7, 1, ' Volume ');
    position_text (65, 16, ' Year ');
    write (end_reverse_video);
    position_text (22, 16, '1981');
    position_text (38, 16, '1982');
    position_text (54, 16, '1983');
    position_text (10, 12, '100');
    position_text (10, 8, '200');
    position_text (10, 4, '300');
    write (end_dim);


    { draw borders & dotted lines }

```

Figure B-3

```
     writeln (enter_graphic_mode, 'REPLACE 15 COLOR');
     writeln ('120 450 120 160 LINE 520 160 LINETO');

     writeln ('272 470 0 470 LINE 0 0 LINETO 639 0 LINETO');
     writeln ('639 470 LINETO 400 470 LINETO');

     writeln ('17476 LSTYLE');          { select dotted lines }
     y := 230;
     REPEAT
          writeln (' 120 ', y, ' 520 ', y, ' LINE');
          y := y + 80;
     UNTIL y > 390;
     writeln ('65535 LSTYLE');          { select solid lines }


     { draw graph bars & legend boxes }

     FOR i := 0 TO 2 DO
          draw_boxes (i);


     { write labels on legend boxes (graphic mode text) }

     writeln ('7 COLOR');
     writeln ('144 60 CURRENT " Company X " ');
     writeln ('286 60 CURRENT " Company Y " ');
     writeln ('420 60 CURRENT " Data General " ');

     writeln (leave_graphic_mode);

    END.
```

```
OB generation summary
Total program code size = 881
Total program literal size = 494
Dispatch tables size = 0
Zrel size = 0
Unshared code size = 23
Unlabeled common_size = 0

147 source lines were compiled in 50 seconds

No Compilation Errors
```

Figure B-3

Assembler Language Calls C

This appendix contains information for the programmer who intends to write assembly language software. It describes the EHYP Assembler instruction which activates the monitor functions. For more details on the display hardware, consult the *Model 10 and 10/SP Computer Systems Technical Reference*, DGC no. 014-000766.

The EHYP Assembler instruction directs the system console emulator to display alphanumeric characters, to modify the character attributes, and to draw graphic objects.

The system console emulator must be loaded into memory before executing the EHYP instruction.

The EHYP Assembler instruction sequence directly accesses the routines in the system console emulator. You access a routine by initializing the four accumulators and then executing the EHYP Assembler instruction (op-code 64004_8).

NOTE *Before executing the EHYP instruction, you must have I/O privileges and disable LEF mode. For more information, refer to the 16-Bit Real-Time ECLIPSE Assembly Language Programming manual.*

The format for the EHYP instruction sequence is:

```
EHYP              ; Jump indirect through AC0; bit 0 = 1

error return      ; Continue here if an error is detected
                  ; Error return is also taken if the
                  ; command interpreter is not loaded

normal return     ; Continue here if no error is detected
```

The accumulators must contain the information shown in Table C-1. Upon return from the EHYP instruction, the original contents of AC3 are modified.

NOTE *Execute the EHYP instruction with extreme caution. Improper execution and return of the EHYP instruction can hang the system. If such an error occurs, the system power must be turned off and on, making it necessary to reboot the operating system.*

Table C-1 EHYP accumulator requirements

ACn	Contents
0	100577_8; indirect address to command interpreter
1	A routine number [See Table C-2 for a list of the routine (call) numbers and command names.]
2	Routine-dependent information
3	Reserved

Accumulator 1 (AC1) must contain the routine (call) number shown in Table C-2. The commands are presented in alphabetical order.

Table C-2 EHYP Assembler Calls

Call Number		Command Name	Description
octal	hex		
0	00	COLD	Reset system console to power-up state
1	01	CLEAR	Clear the display
2	02	COLOR	Set the drawing color
3	03	LSTYLE	Set the line style
4	04	FSTYLE	Set the box fill style
5	05	CURRENT	Set the current pixel address
6	06	POINT	Plot a point
7	07	LINE	Draw a line between two explicit points
10	08	"text" or 'text'	Draw a string of text
11	09	FILLBOX	Draw a box and fill it with the FSTYLE
12	0A	VALUE	Return the color of the addressed pixel
13	0B	LINETO	Draw a line between the current point and another explicit point
14	0C	SELPAL	Select one of four color palettes
15	0D	DEFPAL	Define the colors on a palette
16	0E	SET, REPLACE, XOR	Set the current drawing mode
17	0F	Write Character String	Draw a string of text without the attributes
20	10	Write Attribute String	Modify the appearance of a string of text
21	11	Write Character and Attribute String	Draw a string of text with the attributes
22	12	Write DGC-standard Character String	Draw a string of text with DGC-standard attributes
23	13	Write DGC-standard Character	Draw a DGC-standard Character
24	14	Cursor Address Read	Read the column and line position of the cursor
25	15	Define Character	Modify a character definition in the character font table
26	16	BLINK	Enable or disable graphic blinking
27	17	Write IBM-compatible Character and Attribute String	Draw a string of text with the IBM-compatible attributes

Blink

For a monochrome monitor, the Assembler call takes the error return.

For a color monitor, the Assembler call sets the blink flag to true or false as specified in AC2. The flag is true when the value is from 1 to 65535; false when the value is 0. The flag at power up is false.

With the blink flag true, graphic blinking is enabled; the graphic colors 8 through 15 alternate between the graphic foreground and graphic background colors. The graphic colors 0 through 7 are always the foreground color.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-3.

Table C-3 EHYP Blink accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	26_8; routine number
2	Current BLINK flag value
3	Reserved

Example

The following Assembler statements enable graphic blinking.

```
LDA    0,GRAPHE,0    ; GRAPHE contains 100577₈
ELEF   1,26,0        ; Load call number into AC1
ELEF   2,1           ; Load the flag value into AC2
EHYP                 ; Execute call
error return
normal return
```

Clear

The Assembler call clears the display by setting every pixel to the background color as specified in AC2. The current location becomes 0, 0. The specified background color becomes the current graphic background color.

The color range is 00 to 15 of the current palette. The color selected with this call becomes the background color for *BLINK*, *FSTYLE*, *LSTYLE*, and *REPLACE* commands.

For monochrome, a color of 00 (black) or 01 (green) is available; other color selections (02 to 15) are considered to be 01. At power up, the graphic background color is 00.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-4.

Table C-4 EHYP Clear accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
2	2; routine number
2	Background color; 0 to 15
3	Reserved

Example

The following Assembler statements clear the display with the color specified in AC2.

```
LDA   0,GRAPHE,0      ; GRAPHE contains 100577₈
ELEF   1,1,0          ; Load call number into AC1
ELEF   2,1            ; Load the color into AC2
EHYP                  ; Execute call
error return
normal return
```

For a monochrome display, the command erases the display in green.

Cold

The Assembler call resets the system console to its initial power-up state with the command interpreter loaded. The display is cleared, the cursor returns to the home position. The reset conditions listed in the *Reset* alphanumeric command are put into effect.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-5.

Table C-5 EHYP Cold accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	0; routine number
2	not used
3	Reserved

Example

The following Assembler statements reset the system console.

```
LDA   0,GRAPHE,0        ; GRAPHE contains 100577₈
ELEF   1,0,0            ; Load call number into AC1
EHYP                    ; Execute call
error return
normal return
```

Color

The Assembler call selects, as specified in AC2, the current graphic foreground color. The color range is 00 to 15 of the current palette. For monochrome, a color of 00 (black) or 01 (green) is available; other color selections (02 to 15) are considered to be 01. At power up, the graphic foreground color is 2.

The *FSTYLE, LSTYLE, POINT, REPLACE, SET, "text"*, and *XOR* commands refer to the current graphic foreground color.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-6.

Table C-6 EHYP Color accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	2; routine number
2	Foreground color; 0 to 15
3	Reserved

Example

The following Assembler statements set the current foreground color as specified in AC2.

```
LDA   0,GRAPHE,0       ; GRAPHE contains 100577₈
ELEF   1,2,0           ; Load call number into AC1
ELEF   2,15.           ; Load the color into AC2
EHYP                   ; Execute call
error return
normal return
```

The call sets the current foreground color to 15.

Current

The Assembler call sets the current pixel position to the point (x,y) specified in AC2. The x-coordinate can range from 0 to 639. The y-coordinate can range from 0 to 479. The current position at power up is 0,0.

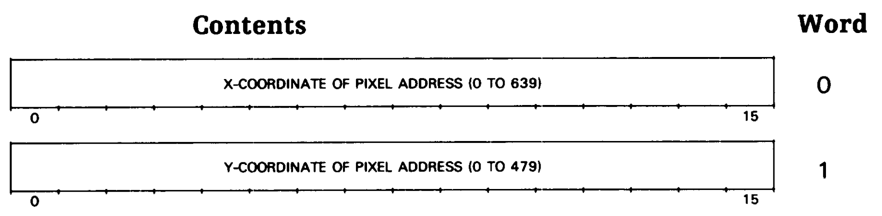
The *LINETO* and *"text"* commands use the current position value.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-7.

Table C-7 EHYP Current accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	5; routine number
2	Pixel packet (word address)
3	Reserved

The pixel packet contains the following 2 words.


Contents
Word
X-COORDINATE OF PIXEL ADDRESS (0 TO 639)
0
15
0
Y-COORDINATE OF PIXEL ADDRESS (0 TO 479)
0
15
1

Example

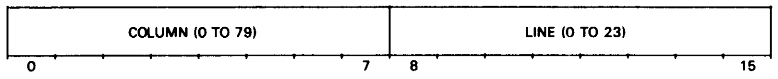
The following Assembler statements define the packet and then call the routine. The call sets the current location to 150,250.

```
PACKET:   150.                 ; x-coordinate
          250.                 ; y-coordinate
          .
          .
          .
          LDA   0,GRAPHE,0     ; GRAPHE contains 100577₈
          ELEF   1,5,0         ; Load call number into AC1
          ELEF   2,PACKET      ; Load packet address into AC2
          EHYP                 ; Execute call
          error return
          normal return
```

The call sets the current pixel position to the 150,250.

Cursor Address Read

The Assembler call returns the cursor location to AC2, where bits 0-7 contain the column number and bits 8-15 contain the line number.



Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-8.

Table C-8 EHYP Cursor Address Read Accumulator Requirements

ACn	Contents
0	100577_8; address to command interpreter
1	24_8; routine number
2	not used
3	Reserved

Example

The following Assembler statements return the <117><000> cursor location to AC2.

```
LDA    0,GRAPHE,0         ; GRAPHE contains 100577₈
ELEF   1,24,0
EHYP
error return
normal return
```

Define Character

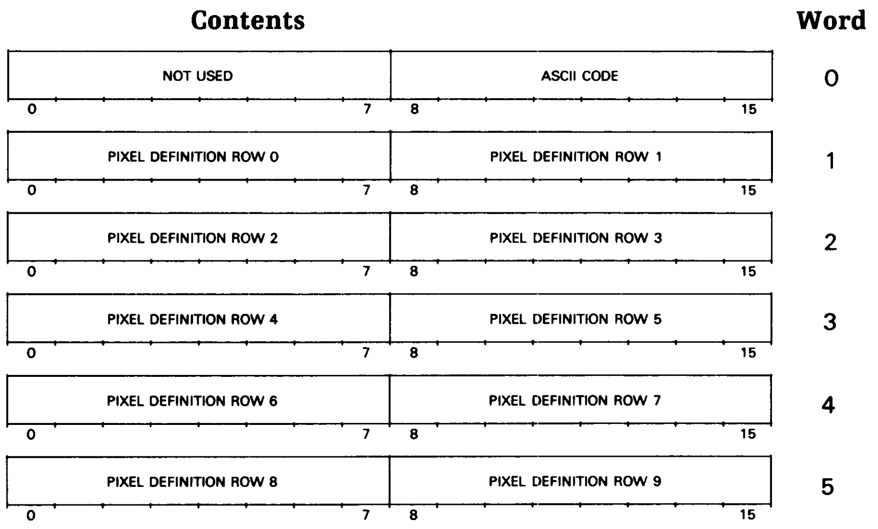
The Assembler call replaces the 8 by 10 pixel matrix that defines the character in the character font table with the replacement string located in a character packet defined by AC2. Thus for the monochrome monitor, the routine changes the appearance of the subsequently drawn characters. For the color monitor, the routine changes the appearance of all the characters.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-9.

Table C-9 EHYP Define Character accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	25_8; routine number
2	Character packet address (word address)
3	Reserved

The character packet contains the following 6 words.



The ASCII code identifies the character whose font is to change. The ASCII codes that can be modified are <040> through <176> and <240> through <376>.

The pixel definition words provide the modified pixel values. The character is defined by 8 pixels horizontally and 10 pixels vertically. Bit 0 of the first row (row 0) is the upper lefthand corner of the character position. Bit 15 of the last row (row 9) is the lower righthand corner of the character position.

Example

The example below shows a font change to the uppercase letter Z. Column (a) shows the character before the change; (b) shows the font specification to change the display; and (c) shows the character after the change.

(a)	(b)	(c)
00000000	00000000	00000000
01111100	01111100	01111100
00000100	00000100	00000100
00001000	00001000	00001000
00010000	01111100	01111100
00100000	00100000	00100000
01000000	01000000	01000000
01111100	01111100	01111100
00000000	00000000	00000000
00000000	00000000	00000000

The following Assembler statements define the packet and then call the routine.

```
         .RDX   8
PACKET:  .TXT   / Z/          ; Change the Z character
         000174               ; Begin pixel definition
         002010
         076040
         040174
         000000
         .
         .
         .
         LDA   0,GRAPHE,0     ; GRAPHE contains 100577₈
         ELEF   1,25,0        ; Load call number into AC1
         ELEF   2,PACKET      ; Load packet address into AC2
         EHYP                 ; Execute call
         error return
         normal return
```

On a monochrome monitor, all future occurrences of the letter Z would be displayed with a bar through the middle of the letter. On a color monitor, all occurrences of the letter Z are displayed with a bar through the middle of the letter.

Define Palette

For a monochrome monitor, the Assembler call takes the error return.

For a color monitor, the Assembler call defines the 16 absolute color values for a graphic palette. The color definition string is located in a palette packet defined by AC2. The value of abs_color ranges from 0 to 4095. The value of color ranges from 00 to 15. The value of palette ranges from 0 to 3.

The color arguments for the *CLEAR*, *COLOR*, and *VALUE* commands refer to this color definition.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-10.

Table C-10 EHYP DEFPAL accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	15_8; routine number
2	Palette packet address (word address)
3	Reserved

The palette packet contains the following 17 words.

	Contents		Word
0	PALETTE NUMBER; 0 TO 3	15	0
0	COLOR 0 DEFINITION; 0 TO 4095	15	1
0	COLOR 1 DEFINITION; 0 TO 4095	15	2
0	COLOR 2 DEFINITION; 0 TO 4095	15	3
0	COLOR 3 DEFINITION; 0 TO 4095	15	4
	⋮		
0	COLOR 14 DEFINITION; 0 TO 4095	15	15
0	COLOR 15 DEFINITION; 0 TO 4095	15	16

The palette number selects the palette to be defined.

A color definition identifies the absolute color for the color position (packet word) on the palette. Referring to the RGB color model, the absolute color can be defined as:

(256*R)+(16*G)+B

where R, G, and B range from 0 to 15.

Example

The following Assembler statements define the packet and then call the routine.

```
PACKET:   1                    ; Select palette 1
          0000.                ; Begin abs_color definitions
          2000.                ; Color 1 is 2000
          0320.                ; Color 2 is 0320
          0174.                ; Color 3 is 0174
          .
          .
          .
          3004.                ; Color 14 is 3004
          4095.                ; Color 15 is 4095
          .
          .
          .
          LDA  0,GRAPHE,0      ; GRAPHE contains 100577₈
          ELEF  1,15,0         ; Load call number into AC1
          ELEF  2,PACKET       ; Load packet address into AC2
          EHYP                 ; Execute call
          error return
          normal return
```

The Assembler call defines all 16 colors on palette 1.

EXCLUSIVE OR Mode

The Assembler call selects the "exclusive or" (XOR) graphic mode, which affects the coloring of pixels for box, line, point, and text operations. The pixels affected are those pixels identified by the *FSTYLE*, *LSTYLE*, *POINT*, and *"text"* commands. The XOR mode is useful for drawing an application-defined graphic cursor.

When the value of a pixel is 1, then the color ordinal of the pixel is exclusively colored with the current foreground color ordinal. When the value of a pixel is 0, then the pixel is not changed. Refer to Appendix B for other examples of the *XOR* command.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-11.

Table C-11 EHYP XOR mode accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	16_8; routine number
2	Contains: 0 for REPLACE mode 1 for SET mode 2 for XOR mode
3	Reserved

Example

The following Assembler statements call the routine.

```
LDA   0,GRAPHE,0      ; GRAPHE contains 100577₈
ELEF   1,16,0         ; Load call number into AC1
ELEF   2,2            ; Load XOR mode into AC2
EHYP                  ; Execute call
error return
normal return
```

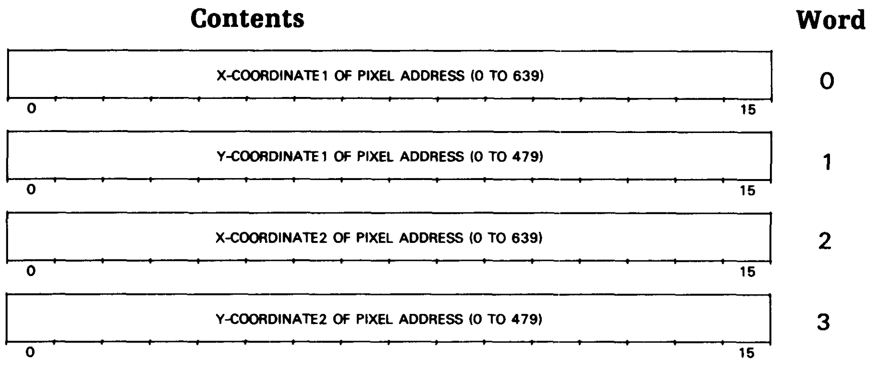
The XOR graphic mode is selected.

Fillbox

The Assembler call draws a filled rectangle as defined by the opposite corners x1,y1 and x2,y2. The x-coordinates can range from 0 to 639. The y-coordinates can range from 0 to 479. The box is filled in the current fill style (*FSTYLE*), graphic mode, and color (*COLOR*). Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-12.

Table C-12 EHYP Fillbox accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	11_8; routine number
2	Pixel packet address (word address)
3	Reserved


Contents
Word
X-COORDINATE1 OF PIXEL ADDRESS (0 TO 639)
0
0
15
Y-COORDINATE1 OF PIXEL ADDRESS (0 TO 479)
1
0
15
X-COORDINATE2 OF PIXEL ADDRESS (0 TO 639)
2
0
15
Y-COORDINATE2 OF PIXEL ADDRESS (0 TO 479)
3
0
15

Example

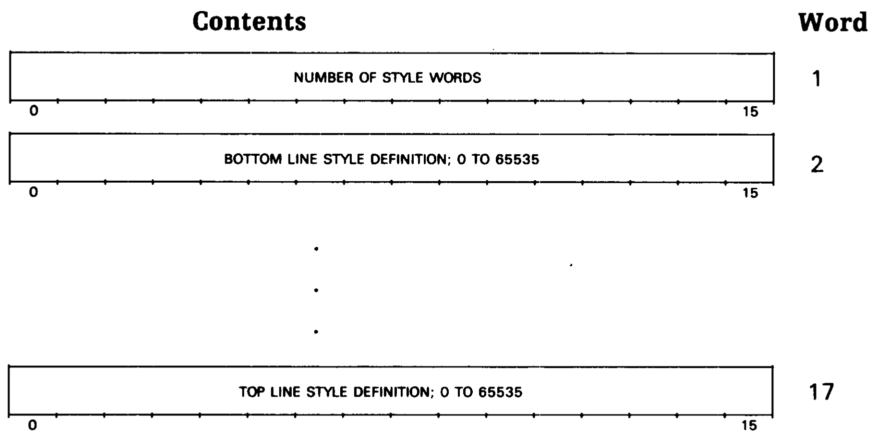
The following Assembler statements define the packet and then call the routine.

```
PACKET:  500.
         100.
         120.
         50.
         .
         .
         .
         LDA   0,GRAPHE,0     ; GRAPHE contains 100577₈
         ELEF  1,11,0         ; Load call number into AC1
         ELEF  2,PACKET       ; Load packet address into AC2
         EHYP                 ; Execute call
         error return
         normal return
```

The call draws a filled rectangle as defined by the corners 500,100 and 120,50.

Fstyle

The Assembler call sets the current fill style for the *FILLBOX* command to the specified style. The fill style packet contains from 2 to 17 words. Word 1 contains the number of words that define the fill style. Word 2 (minimum number of words) through word 17 (maximum number of words) contain a line style in the range 0 to 65535. Each word defines 16 horizontal (x axis) pixels.


Contents
Word
NUMBER OF STYLE WORDS
0
15
1
BOTTOM LINE STYLE DEFINITION; 0 TO 65535
0
15
2
TOP LINE STYLE DEFINITION; 0 TO 65535
0
15
17

The command interpreter fills a box beginning at the lower-lefthand corner of the box with bit 0 of the second word. The next word defines the next higher y axis (another 16 horizontal pixels). When the y axis of a box is larger than the number of words in the fill style definition, the style repeats beginning with word 2. The default style at power up is 65535 (solid box).

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-13.

Table C-13 EHYP Fstyle accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	4; routine number
2	Fill style packet address (word address)
3	Reserved

Example

The following Assembler statements set the current fill style as specified in AC2.

```
PACKET:   00016.             ; Word 1 - number of fill style words
          61680.             ; Word 2 of fill style
          61680.             ; Word 3 of fill style
          .
          .
          .
          LDA   0,GRAPHE,0   ; GRAPHE contains 100577₈
          ELEF  1,4,0        ; Load call number into AC1
          ELEF  2,PACKET     ; Load packet address into AC2
          EHYP               ; Execute call
          error return
          normal return
```

The call produces a striped box.

Line

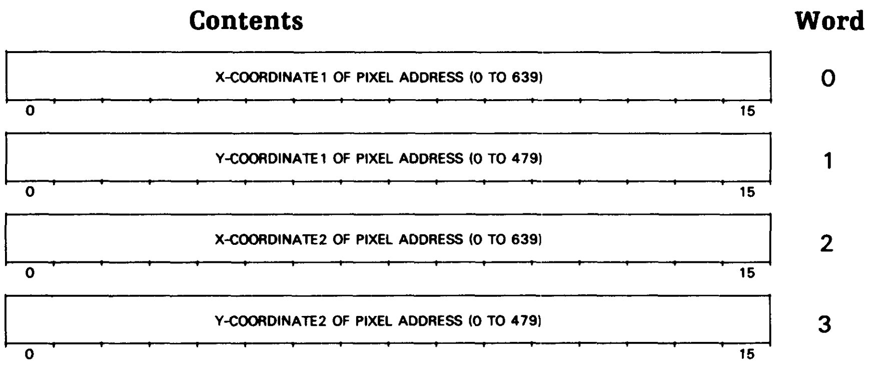
The Assembler call draws a line segment between two specified locations (x1,y1 and x2,y2) in the current foreground color, graphic mode, and line style. The x-coordinate can range from 0 to 639. The y-coordinate can range from 0 to 479. After the line segment is drawn, the coordinates of the end point become the current location.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-14.

Table C-14 EHYP Line accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	7; routine number
2	Pixel packet (word address)
3	Reserved

The pixel packet contains the following 4 words.


Contents
Word
X-COORDINATE1 OF PIXEL ADDRESS (0 TO 639)
0
0
15
Y-COORDINATE1 OF PIXEL ADDRESS (0 TO 479)
1
0
15
X-COORDINATE2 OF PIXEL ADDRESS (0 TO 639)
2
0
15
Y-COORDINATE2 OF PIXEL ADDRESS (0 TO 479)
3
0
15

Example

The following Assembler statements define the packet and then call the routine.

```
PACKET:   100.                      ; x1 coordinate
          100.                      ; y1 coordinate
          200.                      ; x2 coordinate
          100.                      ; y2 coordinate
          .
          .
          .
          LDA    0,GRAPHE,0         ; GRAPHE contains 100577₈
          ELEF    1,7,0             ; Load call number into AC1
          ELEF   2,PACKET           ; Load packet address into AC2
          EHYP                      ; Execute call
          error return
          normal return
```

The call draws a line segment between the two specified coordinates 100,100 and 200,100.

Lineto

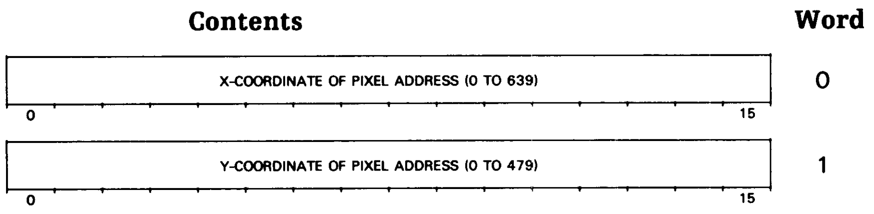
The Assembler call draws a line segment between two specified locations (the current position and x2,y2) in the current foreground color, graphic mode, and line style. The x-coordinate can range from 0 to 639. The y-coordinate can range from 0 to 479. After the line segment is drawn, the coordinates of the end point become the current location.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-15.

Table C-15 EHYP Lineto accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	13_8; routine number
2	Pixel packet (word address)
3	Reserved

The pixel packet contains the following 2 words.


Contents
Word
X-COORDINATE OF PIXEL ADDRESS (0 TO 639)
0
15
0
Y-COORDINATE OF PIXEL ADDRESS (0 TO 479)
0
15
1

Example

The following Assembler statements define the packet and then call the routine.

```
PACKET:  200.                  ; x-coordinate
         100.                  ; y-coordinate
         .
         .
         .
         LDA   0,GRAPHE,0      ; GRAPHE contains 100577₈
         ELEF   1,13,0         ; Load call number into AC1
         ELEF   2,PACKET       ; Load packet address into AC2
         EHYP                  ; Execute call
         error return
         normal return
```

The call draws a line segment between the two specified coordinates 200,100 and the current coordinate.

Lstyle

The Assembler call sets to the specified style (0 to 65535) the current line style for the *LINE* and *LINETO* commands. The style definition contains 16 bits (numbered 0 to 15); one bit for each one of the 16 pixels. A line segment begins with bit 0 of the style definition. The default style at power up is 65535.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-16.

Table C-16 EHYP Lstyle accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	3; routine number
2	Current line style; 0 to 65535
3	Reserved

Example

The following Assembler statements set the current line style as specified in AC2.

```
LDA   0,GRAPHE,0      ; GRAPHE contains 100577₈
ELEF   1,3,0          ; Load call number into AC1
ELEF  2,61680.        ; Load fill style into AC2
EHYP                  ; Execute call
error return
normal return
```

The call produces a dashed line, where alternating 4 pixels contain the current foreground color and the current background color.

Point

The Assembler call draws a pixel at the specified location (x,y). The pixel is painted with current foreground color and graphic mode. The x-coordinate can range from 0 to 639. The y-coordinate can range from 0 to 479. After the point is drawn, the coordinates become the current location.

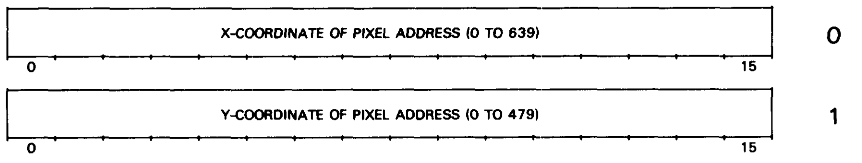
Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-17.

Table C-17 EHYP Point accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	6; routine number
2	Pixel packet (word address)
3	Reserved

The pixel packet contains the following 2 words.

Contents **Word**


X-COORDINATE OF PIXEL ADDRESS (0 TO 639)
0
15
0
Y-COORDINATE OF PIXEL ADDRESS (0 TO 479)
0
15
1

Example

The following Assembler statements define the packet and then call the routine. The call draws one point at 200,100.

```
PACKET:   200.                 ; x-coordinate
          100.                 ; y-coordinate
          .
          .
          .
          LDA   0,GRAPHE,0     ; GRAPHE contains 100577₈
          ELEF   1,6,0         ; Load call number into AC1
          ELEF   2,PACKET      ; Load packet address into AC2
          EHYP                 ; Execute call
          error return
          normal return
```

The call draws a pixel at 200,100.

Replace Mode

The Assembler call selects the REPLACE graphic mode, which affects the coloring of pixels for box, line, point, and graphic text operations. The pixels affected are those pixels identified by the *FSTYLE*, *LSTYLE*, *POINT*, and *"text"* commands. The REPLACE mode is useful for drawing an object (or graphic text) over an existing object, where portions of the existing object become invisible.

When the value of a pixel is 1, then the pixel is set to the current foreground color. When the value of a pixel is 0, then the pixel is set to the current background color. Refer to Appendix B for an example that further clarifies the effects of the *REPLACE* command.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-18.

Table C-18 EHYP Replace Mode accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	16_8; routine number
2	Contains: 0 for REPLACE mode 1 for SET mode 2 for XOR mode
3	Reserved

Example

The following Assembler statements call the routine.

```
LDA    0,GRAPHE,0       ; GRAPHE contains 100577₈
ELEF    1,16,0          ; Load call number into AC1
ELEF   2,0              ; Load REPLACE mode into AC2
EHYP                    ; Execute call
error return
normal return
```

The REPLACE graphic mode is selected.

Select Graphic Palette

For a monochrome monitor, the Assembler call takes the error return.

For a color monitor, the Assembler call defines the current palette (palette). The value of palette ranges from 0 to 3. At power up, the current palette is 0.

The color arguments for the *CLEAR*, *COLOR*, *POINT*, and *VALUE* commands access the current palette.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-19.

Table C-19 EHYP SELPAL accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	14_8; routine number
2	Palette number; 0 to 3
3	Reserved

Example

The following Assembler statements call the routine.

```
LDA   0,GRAPHE,0      ; GRAPHE contains 100577₈
ELEF   1,14,0         ; Load call number into AC1
ELEF   2,1            ; Load palette 1 into AC2
EHYP                  ; Execute call
error return
normal return
```

The call selects palette 1.

Set Mode

The Assembler call selects the SET graphic mode, which affects the coloring of pixels for box, line, point, and text operations. The pixels affected are those pixels identified by the *FSTYLE*, *LSTYLE*, *POINT*, and *"text"* commands. The SET mode is useful for drawing an object (or graphic text) on top of another object, where portions of both objects are visible.

When the value of a pixel is 1, then the pixel is set to the current foreground color. When the value of a pixel is 0, then the pixel is not changed. Refer to Appendix B for an example that further clarifies the effects of the *SET* command.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-20.

Table C-20 EHYP Set Mode accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	16_8; routine number
2	Contains: 0 for REPLACE mode 1 for SET mode 2 for XOR mode
3	Reserved

Example

The following Assembler statements call the routine.

```
LDA   0,GRAPHE,0      ; GRAPHE contains 100577₈
ELEF   1,16,0         ; Load call number into AC1
ELEF   2,1            ; Load SET mode into AC2
EHYP                  ; Execute call
error return
normal return
```

The SET graphic mode is selected.

Text String

The Assembler call draws a graphic text string. The string then begins with the lower-left corner of the first character at the current location. The string as sent to the command interpreter must begin and end with a set of double quote marks (") or a set of single marks (').

The command interpreter draws the string in the current graphic foreground color and graphic mode. All the display characters can be drawn with this command. Control characters are ignored.

The string can be 1 to 80 characters. If more than 80 characters are specified or the drawing reaches the display border, the extra characters are ignored.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-21.

Table C-21 EHYP LSTYLE accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	10_8; routine number
2	Pixel packet (word address)
3	Reserved

The pixel packet contains the following 2 words.

Contents (bits 0–15)	Word
NUMBER OF CHARACTERS TO DRAW (1 TO 80)	0
BYTE POINTER TO FIRST CHARACTER	1

Example

The following Assembler statements define the packet and then call the routine.

```
PACKET:    22.
           MESS.2
MESS       .TXT   /A SAMPLE TEXT
           MESSAGE./
           .
           .
           .
           LDA   0,GRAPHE,0        ; GRAPHE contains 100577₈
           ELEF   1,10,0           ; Load call number into AC1
           ELEF   2,PACKET         ; Load packet address into AC2
           EHYP                    ; Execute call
           error return
           normal return
```

The call draws a graphic text string, 'A sample text message.'.

Value

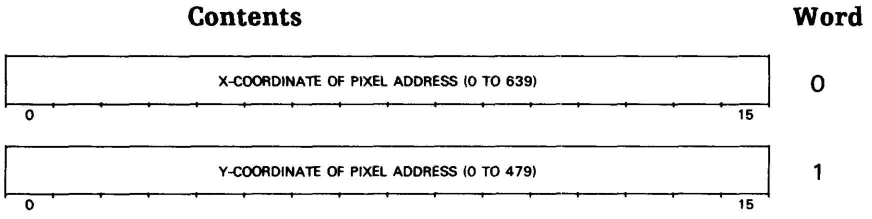
The Assembler call returns the color value to AC2. The x- and y-coordinates are specified in a pixel packet. The x-coordinate ranges from 0 to 639. The y-coordinate ranges from 0 to 479. The value, ranging from 00 to 15, represents a graphic color from the currently selected palette.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-22.

Table C-22 EHYP Value accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	12_8; routine number
2	Pixel packet (word address)
3	Reserved

The pixel packet contains the following 2 words.


Contents
Word
X-COORDINATE OF PIXEL ADDRESS (0 TO 639)
0
15
0
Y-COORDINATE OF PIXEL ADDRESS (0 TO 479)
0
15
1

Example

The following Assembler statements define the packet and then call the routine.

```
PACKET:   50.                ; x-coordinate
          100.               ; y-coordinate
          .
          .
          .
          LDA   0,GRAPHE,0   ; GRAPHE contains 100577₈
          ELEF   1,12,0      ; Load call number into AC1
          ELEF   2,PACKET    ; Load packet address into AC2
          EHYP               ; Execute call
          error return
          normal return
```

The Assembler call returns in AC2 the color value of the pixel. For example if the pixel addressed at 50,100 contains the color value 15, then AC2 contains 17_8.

Write Attribute String

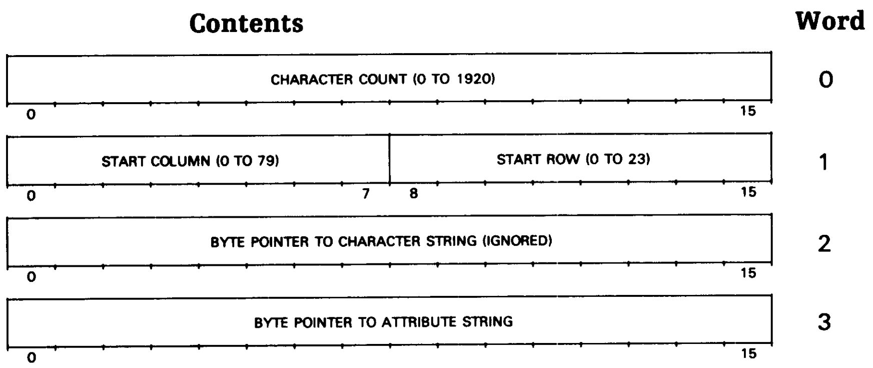
The Assembler call defines or modifies the character attribute table for an alphanumeric character string. The call modifies the character string, which begins at the specified column and line address. The cursor location is not changed.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-23.

Table C-23 EHYP Write Attribute String accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	20_8; routine number
2	String packet address (word address)
3	Reserved

The string packet contains the following 4 words.


Contents
Word
CHARACTER COUNT (0 TO 1920)
0
15
0
START COLUMN (0 TO 79)
START ROW (0 TO 23)
0
7
8
15
1
BYTE POINTER TO CHARACTER STRING (IGNORED)
0
15
2
BYTE POINTER TO ATTRIBUTE STRING
0
15
3

The character count, ranging from 0 to 1920, specifies the number of character attributes to modify. If the character count exceeds 1920, it is truncated to 1920.

If the character count extends beyond the end of the screen, all data beyond the last character on the screen is ignored.

The first character position, in word 1, is a specified column and line number on the display.

The byte pointer, in word 2, is a byte address to the first character of the string in memory and is ignored.

The byte pointer, in word 3, is a byte address to the attribute string in memory. The attribute string contains 2 bytes for each character. The most significant byte contains the current alphanumeric character foreground and background colors and two of the standard alphanumeric character attributes. The least significant byte contains the remaining character attributes. (The monochrome monitor uses only the fBL, f_I, US, and RV bits, which are the standard alphanumeric character attributes.) The 2 bytes are formatted as follows:

FBL	B_R	B_G	B_B	F_I	F_R	F_G	F_B	US	RV	RESERVED
0	1	2	3	4	5	6	7	8	9	10 15

AC	Description	Function
fBL	Foreground Blink (color and monochrome)	The fBL bit when set to one selects blinking the foreground color.
b_R	Background Red (color)	The b_R bit when set to one selects background red.
b_G	Background Green (color)	The b_G bit when set to one selects background green.
b_B	Background Blue (color)	The b_B bit when set to one selects background blue.
f_I	Foreground Intensity (color and monochrome)	The f_I bit when set to one selects foreground intensity. The foreground color is mixed with white for a lighter shade of the color.
f_R	Foreground Red (color)	The f_R bit when set to one selects foreground red.
f_G	Foreground Green (color)	The f_G bit when set to one selects foreground green.
f_B	Foreground Blue (color)	The f_B bit when set to one selects foreground blue.
US	Underscore (color and monochrome)	The US bit when set to one selects underscoring.
RV	Reverse Video (color and monochrome)	The RV bit when set to one selects reverse video. With reverse video, the foreground and background colors are reversed.
Reserved	Not used	The reserved bits are ignored.

Example

The following Assembler statements define the packet and then call the routine.

```
PACKET:   12.               ; 12 characters in message
          0                 ; Begin message at home
          MESS*2            ; Byte pointer to the character ; string (ignored)
          ATMES*2           ; Byte pointer to the attribute ; string
                            ;
ATMES:    00200             ; Attribute string (of letter T)
          00200             ; Attribute string    .   e
          00200             ; Attribute string    .   s
          00200             ; Attribute string    .   t
          00200             ; Attribute string    .
          00200             ; Attribute string    .   m
          00200             ; Attribute string    .   e
          00200             ; Attribute string    .   s
          00200             ; Attribute string    .   s
          00200             ; Attribute string    .   a
          00200             ; Attribute string    .   g
          00200             ; Attribute string (of letter e)
          .
          .
          .
          LDA  0,GRAPHE,0   ; GRAPHE contains 100577₈
          ELEF  1,20,0      ; Load call number into AC1
          ELEF    2,PACKET  ; Load packet address into AC2
          EHYP              ; Execute call
          error return
          normal return
```

The alphanumeric character attributes are changed to underscore, from line 0 and column 0 to column 11.

Write Character String

The Assembler call displays an alphanumeric character string with whatever attributes are currently defined. The character string can contain any of the display characters. If a control character is specified in the string, it is ignored.

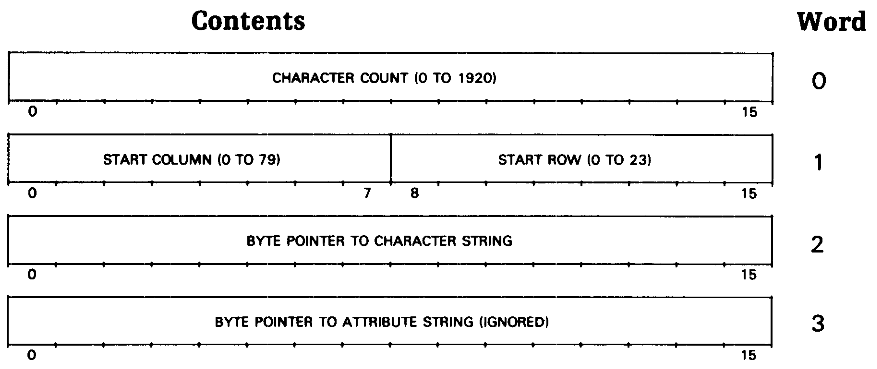
The call displays the character string beginning at the specified column and line address. The cursor location is not modified.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-24.

Table C-24 EHYP Write Character String accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	17_8; routine number
2	String packet address (word address)
3	Reserved

The string packet contains the following 4 words.


Contents
Word
CHARACTER COUNT (0 TO 1920)
0
0
15
START COLUMN (0 TO 79)
START ROW (0 TO 23)
1
0
7
8
15
BYTE POINTER TO CHARACTER STRING
2
0
15
BYTE POINTER TO ATTRIBUTE STRING (IGNORED)
3
0
15

The character count, ranging from 0 to 1920, specifies the number of characters to display. If the character count exceeds 1920, it is truncated to 1920.

If the character count extends beyond the end of the screen, all data beyond the last character on the screen is ignored.

The first character position in word 1 is a specified column and line number on the display.

The byte pointer in word 2 is a byte address to the first character of the string in memory.

The byte pointer to the attribute string in word 3 is ignored.

Example

The following Assembler statements define the packet and then call the routine.

```
PACKET:   12.                     ; 12 characters in message
          0                       ; Begin message at home
          MESS*2                  ; Byte pointer to the character ; string
          ATMES*2                 ; Byte pointer to the attribute ; string (ignored)
                                  ;
MESS:     .TXT   /Test message/   ; Message
          .
          .
          .
          LDA   0,GRAPHE,0        ; GRAPHE contains 100577₈
          ELEF   1,17,0           ; Load call number into AC1
          ELEF      2,PACKET      ; Load packet address into AC2
          EHYP                    ; Execute call
          error return
          normal return
```

The alphanumeric character string is displayed from line 0 and column 0 to column 11. The characters are displayed with the currently defined attributes.

Write Character and Attribute String

The Assembler call displays an alphanumeric character string and its attributes. The character string can contain any of the display characters. If a control character is specified in the string, it is ignored.

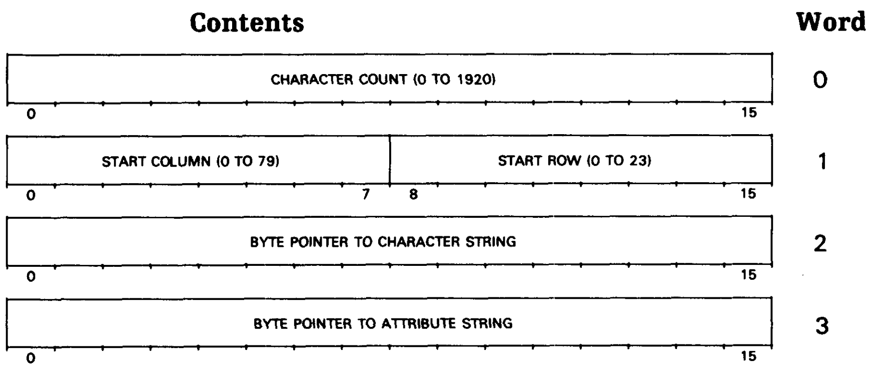
The call displays the string beginning at the specified column and line address. The cursor location is not modified.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-25.

Table C-25 EHYP Write Character and Attribute String accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	21_8; routine number
2	String packet address (word address)
3	Reserved

The string packet contains the following 4 words.


Contents
Word
CHARACTER COUNT (0 TO 1920)
0
15
0
START COLUMN (0 TO 79)
START ROW (0 TO 23)
0
7
8
15
1
BYTE POINTER TO CHARACTER STRING
0
15
2
BYTE POINTER TO ATTRIBUTE STRING
0
15
3

The character count, ranging from 0 to 1920, specifies the number of characters to display. If the character count exceeds 1920, it is truncated to 1920.

If the character count extends beyond the end of the screen, all data beyond the last character on the screen is ignored.

The first character position, in word 1, is a specified column and line number on the display.

The byte pointer, in word 2, is a byte address to the first character of the string in memory.

The byte pointer, in word 3, is a byte address to the attribute string in memory. The attribute string contains 2 bytes for each character. The most significant byte contains the current alphanumeric character foreground and background colors and two of the standard alphanumeric character attributes. The least significant byte contains the remaining character attributes. (The monochrome monitor uses only the fBL, f_I, US, and RV bits, which are the standard alphanumeric character attributes.) The 2 bytes are formatted as follows:

fBL	b_R	b_G	b_B	f_I	f_R	f_G	f_B	US	RV	RESERVED
0	1	2	3	4	5 10	11	12	13	14	15 20

AC	Description	Function
fBL	Foreground Blink (color and monochrome)	The fBL bit when set to one selects blinking the foreground color.
b_R	Background Red (color)	The b_R bit when set to one selects background red.
b_G	Background Green (color)	The b_G bit when set to one selects background green.
b_B	Background Blue (color)	The b_B bit when set to one selects background blue.
f_I	Foreground Intensity (color and monochrome)	The f_I bit when set to one selects foreground intensity. The foreground color is mixed with white for a lighter shade of the color.
f_R	Foreground Red (color)	The f_R bit when set to one selects foreground red.
f_G	Foreground Green (color)	The f_G bit when set to one selects foreground green.
f_B	Foreground Blue (color)	The f_B bit when set to one selects foreground blue.
US	Underscore (color and monochrome)	The US bit when set to one selects underscoring.
RV	Reverse Video (color and monochrome)	The RV bit when set to one selects reverse video. With reverse video, the foreground and background colors are reversed.
Reserved	Not used	The reserved bits are ignored.

Example

The following Assembler statements define the packet and then call the routine.

```
PACKET:   12.                  ; 12 characters in message
          0                    ; Begin message at home
          MESS*2               ; Byte pointer to the character ; string (ignored)
          ATMES*2              ; Byte pointer to the attribute ; string
                               ;
ATMES:    00200                ; Attribute string (of letter T)
          00200                ; Attribute string     .     e
          00200                ; Attribute string     .     s
          00200                ; Attribute string     .     t
          00200                ; Attribute string     .
          00200                ; Attribute string     .     m
          00200                ; Attribute string     .     e
          00200                ; Attribute string     .     s
          00200                ; Attribute string     .     s
          00200                ; Attribute string     .     a
          00200                ; Attribute string     .     g
          00200                ; Attribute string (of letter e)
          .
          .
          .
          LDA   0,GRAPHE,0     ; GRAPHE contains 100577₈
          ELEF   1,20,0        ; Load call number into AC1
          ELEF      2,PACKET   ; Load packet address into AC2
          EHYP                 ; Execute call
          error return
          normal return
```

The alphanumeric character string is displayed, with underscore, from line 0 and column 0 to column 11.

Write DGC-standard Character

The Assembler call writes an alphanumeric character at the cursor location. The cursor location is then updated. The character can be one of the displayable characters or an alphanumeric control (character) command. Undefined control characters are ignored.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-26.

Table C-26 EHYP Write DGC-standard Character accumulator requirements

ACn	Contents
0	100577_8; address to command interpreter
1	23_8; routine number
2	Ignore bits 0 to 7; display the character from bits 8 to 15
3	Reserved

Example

The following Assembler statements display the letter A and updates the cursor location.

```
CHARA:      .TXT   / A/
             .
             .
             .
             LDA   0,GRAPHE,0       ; GRAPHE contains 100577₈
             ELEF   1,23,0          ; Load call number into AC1
             LDA   2,CHARA          ; Load character "A" into AC2
             EHYP                   ; Execute call
             error return
             normal return
```

The call writes the alphanumeric character, 'A'.

Write DGC-standard Character String

The Assembler call displays an alphanumeric character string. The character string can contain any of the display and control characters. Undefined control characters are ignored.

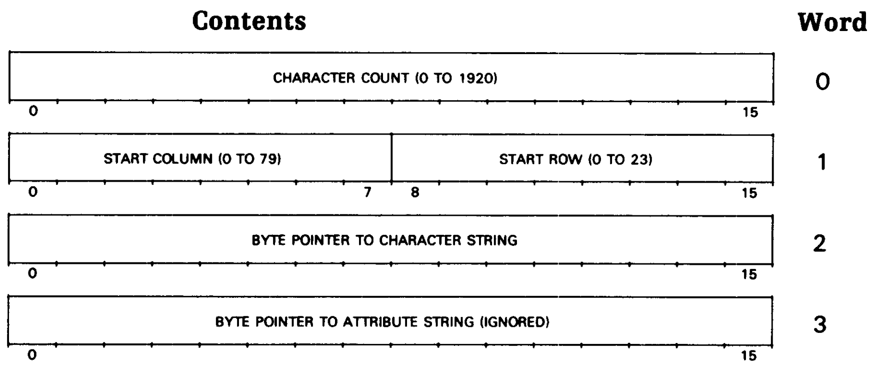
Depending upon the contents of word 1 of the packet, the call displays the character string either beginning at the current cursor position or beginning at the specified column and line address. The cursor location is then updated.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-27.

ACn	Contents
0	100577_8; address to command interpreter
1	22_8; routine number
2	String packet address (word address)
3	Reserved

Table C-27. EHYP write DGC-standard character string accumulator requirements

The string packet contains the following 4 words.


Contents
Word
CHARACTER COUNT (0 TO 1920)
0
0
15
START COLUMN (0 TO 79)
START ROW (0 TO 23)
1
0
7
8
15
BYTE POINTER TO CHARACTER STRING
2
0
15
BYTE POINTER TO ATTRIBUTE STRING (IGNORED)
3
0
15

The character count, ranging from 0 to 1920, specifies the number of characters to display. If the character count exceeds 1920, it is truncated to 1920.

The first character position, in word 1, can be a cursor position or a specified column and line number. If the contents of word 1 equals -1 (177777_8), the first character position is the cursor location. Otherwise, the first character position is the column and line number specified in word 1.

The byte pointer to the character string is a byte address to the first character of the string.

The byte pointer to the attribute string in word 3 is ignored.

Example

The following Assembler statements define the packet and then call the routine.

```
PACKET:   12.                   ; 12 characters in message
          0                     ; Begin message at home
          MESS*2                ; Byte pointer to the character
                                ; string
          ATMES*2               ; Byte pointer to the attribute
                                ; string (ignored)
                                ;
MESS:     .TXT   /Test message/ ; Message
ATMES:

          .
          .
          .

          LDA    0,GRAPHE,0     ; GRAPHE contains 100577₈
          ELEF   1,22,0         ; Load call number into AC1
          ELEF   2,PACKET       ; Load packet address into AC2
          EHYP                  ; Execute call
          error return
          normal return
```

The alphanumeric character string is displayed from line 0 and column 0 to column 11. The cursor location is updated to line 0 and column 12.

Write IBM-compatible Character and Attribute String

The Assembler call displays an alphanumeric character string and its attributes. The character string can contain any of the display characters. If a control character is specified in the string, it is ignored.

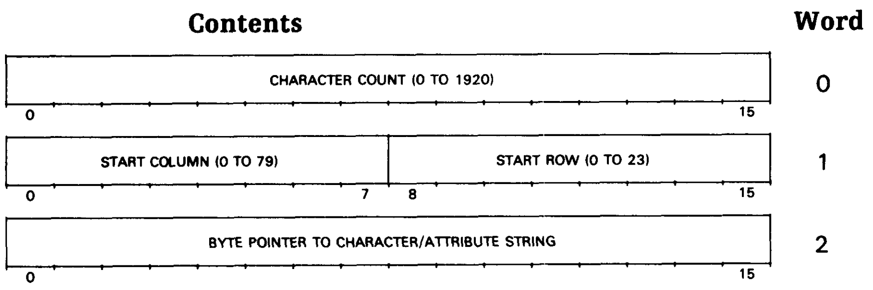
The call displays the string beginning at the specified column and line address. The cursor location is not modified.

Before executing the EHYP instruction, the accumulators must contain the information shown in Table C-28.

ACn	Contents
0	100577_8; address to command interpreter
1	27_8; routine number
2	String packet address (word address)
3	Reserved

Table C-28. EHYP write IBM-mode character and attribute string accumulator requirements

The string packet contains the following 3 words.


Contents
Word
CHARACTER COUNT (0 TO 1920)
0
0
15
START COLUMN (0 TO 79)
START ROW (0 TO 23)
1
0
7
8
15
BYTE POINTER TO CHARACTER/ATTRIBUTE STRING
2
0
15

The character count, ranging from 0 to 1920, specifies the number of characters to display. If the character count exceeds 1920, it is truncated to 1920.

The first character position, in word 1, is a specified column and line number on the display.

The byte pointer, in word 2, is a byte address to the first character and attribute in memory. The two character/attribute bytes are compatible to the IBM PC character/attribute word.

EIGHT-BIT CHARACTER	fBL	b_R	b_G	b_B	f_I	f_R	f_G	f_B
0 7	8	9	10	11	12	13	14	15

AC	Description	Function
eight-bit character	ASCII character code (color and monochrome)	The 8-bit ASCII code of the display character is located in bits 0 through 7.
fBL	Foreground Blink (color and monochrome)	The fBL bit when set to one selects blinking the foreground color.
b_R	Background Red (color)	The b_R bit when set to one selects background red.
b_G	Background Green (color)	The b_G bit when set to one selects background green.
b_B	Background Blue (color)	The b_B bit when set to one selects background blue.
f_I	Foreground Intensity (color and monochrome)	The f_I bit when set to one selects foreground intensity. The foreground color is mixed with white for a lighter shade of the color.
f_R	Foreground Red (color)	The f_R bit when set to one selects foreground red.
f_G	Foreground Green (color)	The f_G bit when set to one selects foreground green.
f_B	Foreground Blue (color)	The f_B bit when set to one selects foreground blue.

NOTE *If any character has a blue foreground (color bits 001) and black background (color bits 000), the character is displayed with an underscore.*

Example

The following Assembler statements define the packet and then call the routine.

```
PACKET:   12.                   ; 12 characters in message
          0                     ; Begin message at home
          MESS*2                ; Byte pointer to the message
                                ;
MESS:     (256.* 84.)+200       ; Character/Attribute (letter T)
          (256.*101.)+200       ; Character/Attribute .   e
          (256.*115.)+200       ; Character/Attribute .   s
          (256.*116.)+200       ; Character/Attribute .   t
          (256.* 32.)+200       ; Character/Attribute .
          (256.*109.)+200       ; Character/Attribute .   m
          (256.*101.)+200       ; Character/Attribute .   e
          (256.*115.)+200       ; Character/Attribute .   s
          (256.*115.)+200       ; Character/Attribute .   s
          (256.* 97.)+200       ; Character/Attribute .   a
          (256.*103.)+200       ; Character/Attribute .   g
          (256.*101.)+200       ; Character/Attribute letter e)

          .
          .
          .

          LDA   0,GRAPHE,0      ; GRAPHE contains 100577₈
          ELEF   1,27,0         ; Load call number into AC1
          ELEF   2,PACKET       ; Load packet address into AC2
          EHYP                  ; Execute call
          error return
          normal return
```

The IBM-compatible alphanumeric character string is displayed as a blinking message. The character string begins at line 0 and column 0 and ends at column 11.

Specifications D

Item	Description Monochrome	 Color
Display Characteristics		
Resolution	640 x 240 pixels	640 x 240 pixels
Color	Black and green only	16 out of 4096
Display Technique	Non-interlaced raster scan	Non-interlaced raster scan
Monitor interface	Proprietary	EIA RS-170 compatible
Monitor phosphor	P31 green phosphor	P22 rare earth phosphors
Mechanical		
Unit weight	8.2 kg (18 lbs)	16.8 kg (37 lbs)
Diagonal screen measurement	304.8 mm or 12″	330.2 mm or 13″
Display dimensions	320 mm (12.8″)wide 305 mm(12.2″)deep	330 mm(13.2″)high
Display ac power cable	1.8 m (6.0 ft) 2.5 m (8.2 ft)	
Keyboard		
Dimensions	Height: 30.5 mm (1.2″) Width: 530.8 mm (20.9″) Depth: 198.1 mm (7.8″)	Same
Weight	2.72 kg (5.8 lbs)	Same
Cable	0.5 m (1.5 ft) coiled 1.2 m (3.7 ft) extended	
Electrical		
Supply voltage/ Tolerance/Frequency	103-126V ac, 57-63Hz 198-264V ac, 47-53Hz	102-132V ac,57-63Hz 187-264V ac,47-53Hz
Environmental		
Operating temperature	0 to 43 C (32 to 110 F)	Same
Storage temperature	−40 to 65 C (−40 to 150 F)	Same
Operating humidity	20 to 80%, non-condensing	Same
Storage humidity	10 to 90%, non-condensing	Same
Operating altitude	3048 m (10,000 ft) maximum	Same

Index

moisten & seal

Documentation Comment Form

Manual Title ______________________________

Manual No. ______________________________

Your Name ______________________________

Your Title ______________________________

Company ______________________________

Street ______________________________

City ____________________ State ________ Zip ____________

Please help us improve our future publications by answering the questions below. Use the space provided for your comments. Thank you.

	Yes	No
Is this manual easy to read?	☐	☐
Is it easy to understand?	☐	☐
Are the topics logically organized?	☐	☐
Is the technical information accurate?	☐	☐
Can you easily find what you want?	☐	☐
Does it tell you everything you need to know?	☐	☐
Do the illustrations help you?	☐	☐

If you wish to order manuals, contact your sales representative or dealer.

Comments:

Date

NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 26 SOUTHBORO, MA. 01772

Postage will be paid by addressee:

DataGeneral

ATTN: Design Services (E219)
4400 Computer Drive
Westboro, MA 01581

134-755